Dr. Dobb's Journal

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS

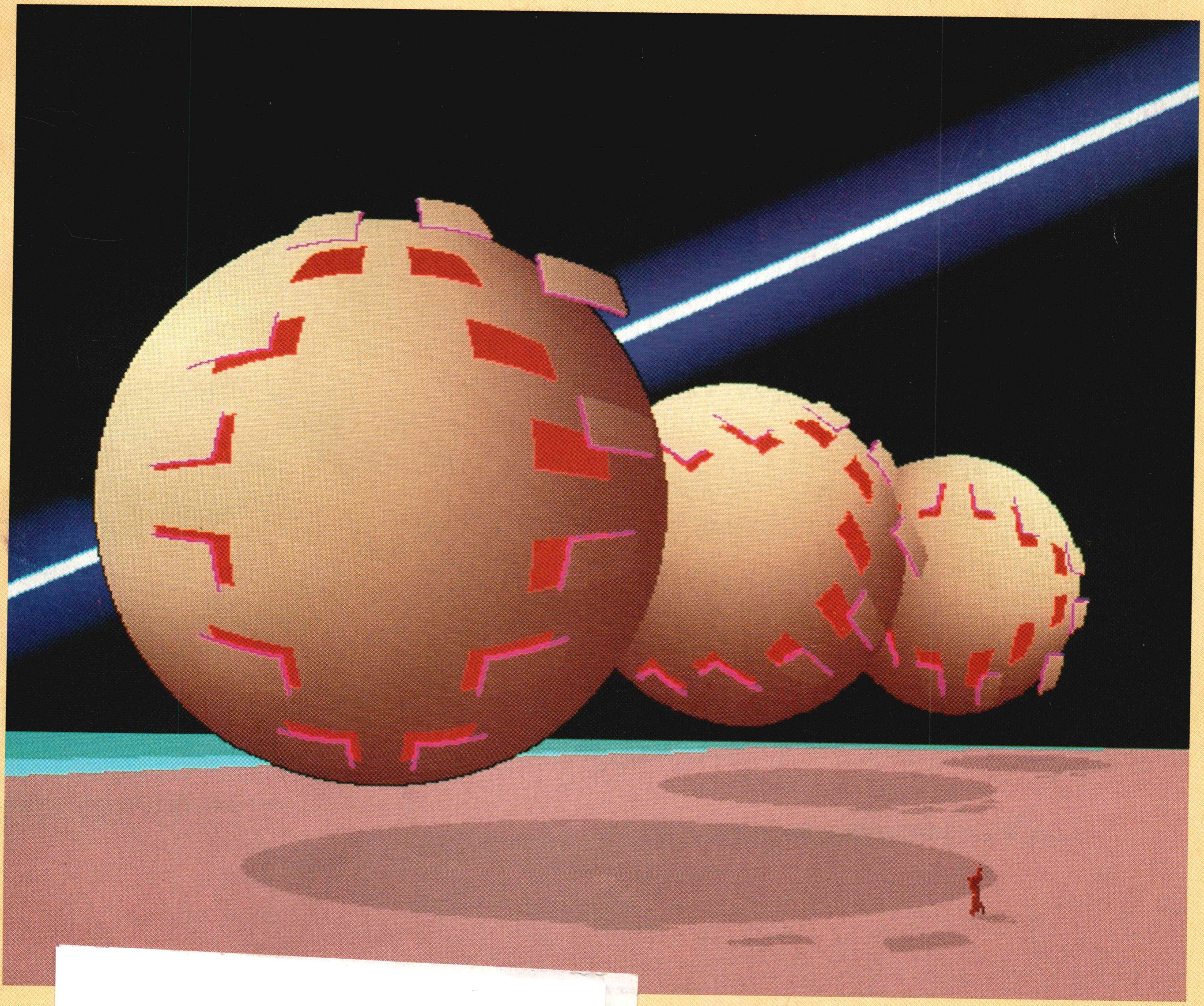
#109 NOVEMBER 1985 $2.95 ($3.95 CANADA)

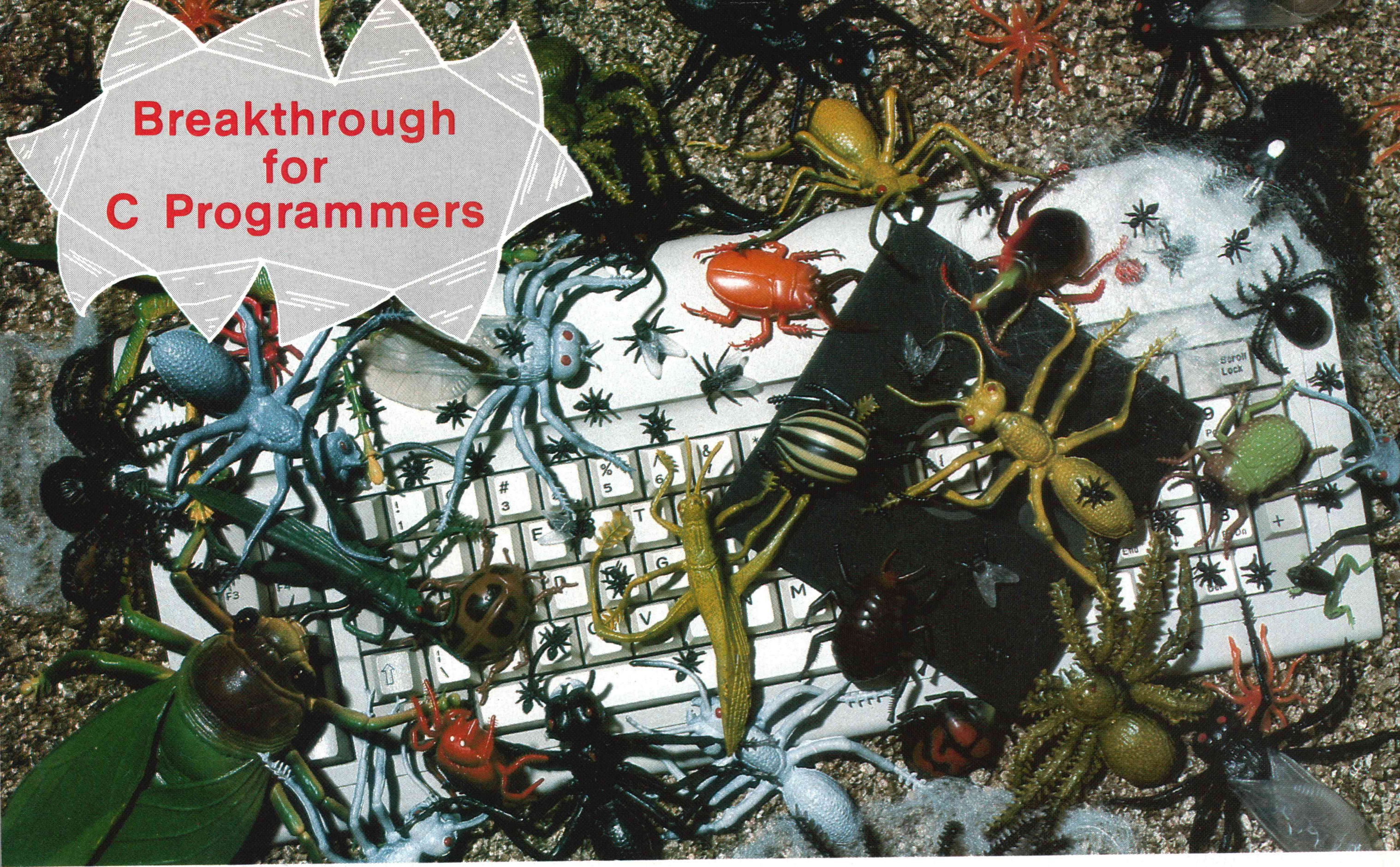
Modula Tools

Programmable Editors

An MSDOS Filter

68K and 286 Tips





H.E.L.P. Eliminates Every Bug known to Compilers
. . . As well as a few other species

H.E.L.P. is a completely interactive C programming environment with three innovative full-sized features that will revolutionize the way you write code.

A Clean Compile — Guaranteed!

Say Good-bye to all compiler-type errors. **H.E.L.P.** 's built-in program checker (which would embarrass LINT) not only hunts down bugs . . . explains the proper syntax . . . gives examples of usage . . . but will even offer suggested corrections. If you want, **H.E.L.P.** will even make the corrections for you . . . at the touch of a key.

H.E.L.P. also finds semantic errors as well as poor style and inefficiencies. You can even check the portability of your code!

Multi-Window Editing

Open as many windows as you want . . . . there's no limitation . . . not even your own memory. Because **H.E.L.P.** uses a virtual memory system you can create programs larger than your machine capacity.

H.E.L.P.'s very powerful editor allows you the flexibility to work in several windows . . . with several files at the same time.

Save Keystrokes

Hundreds of commands are bound to the keyboard to give you fast execution.

Always be in Control

Not only can you develop code in many windows at the same time, but you can show (and refer to) important definitions in one window while creating in another. Or open a window and keep notes about your program . . . or type a memo . . . or a letter.

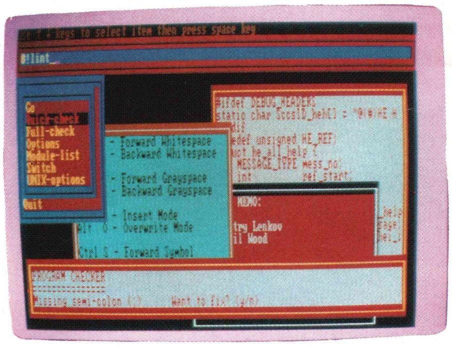
Increase your Productivity by 300% or More . . . . .

If you are a novice programmer, you'll begin writing code like an advanced programmer much faster with **H.E.L.P.**

Just imagine what **H.E.L.P.** will do for the ADVANCED PROGRAMMER.

You'll have more time to become creative with your algorithm (since **H.E.L.P.** will make sure your code compiles the "first time at bat").

H.E.L.P. tracks every step you make. If you are not sure about a command, just press a key, and you'll get the kind of help you need.



Check These Features

- *Multi-window environment*
- *Interactive program checking*
- *Check syntax, semantic, type usage, intermodule inconsistencies and portability*
- *Multi-file editing*
- *Intelligent help sub-system*
- *User-definable key-board bindings*
- *Supports color and monochrome*
- ***H.E.L.P.** supports the full C Language*

NOW IN MS-DOS

EVEREST SOLUTIONS

Order now $395

Everest Solutions, Inc.
3350 Scott Boulevard
Building 58
Santa Clara, CA 95051
(408) 986-8977

Circle **no. 30** on reader service card.

See Us at COMDEX

Optotech, Inc.

The 5¼ inch Optical Disk Drive Is Here!

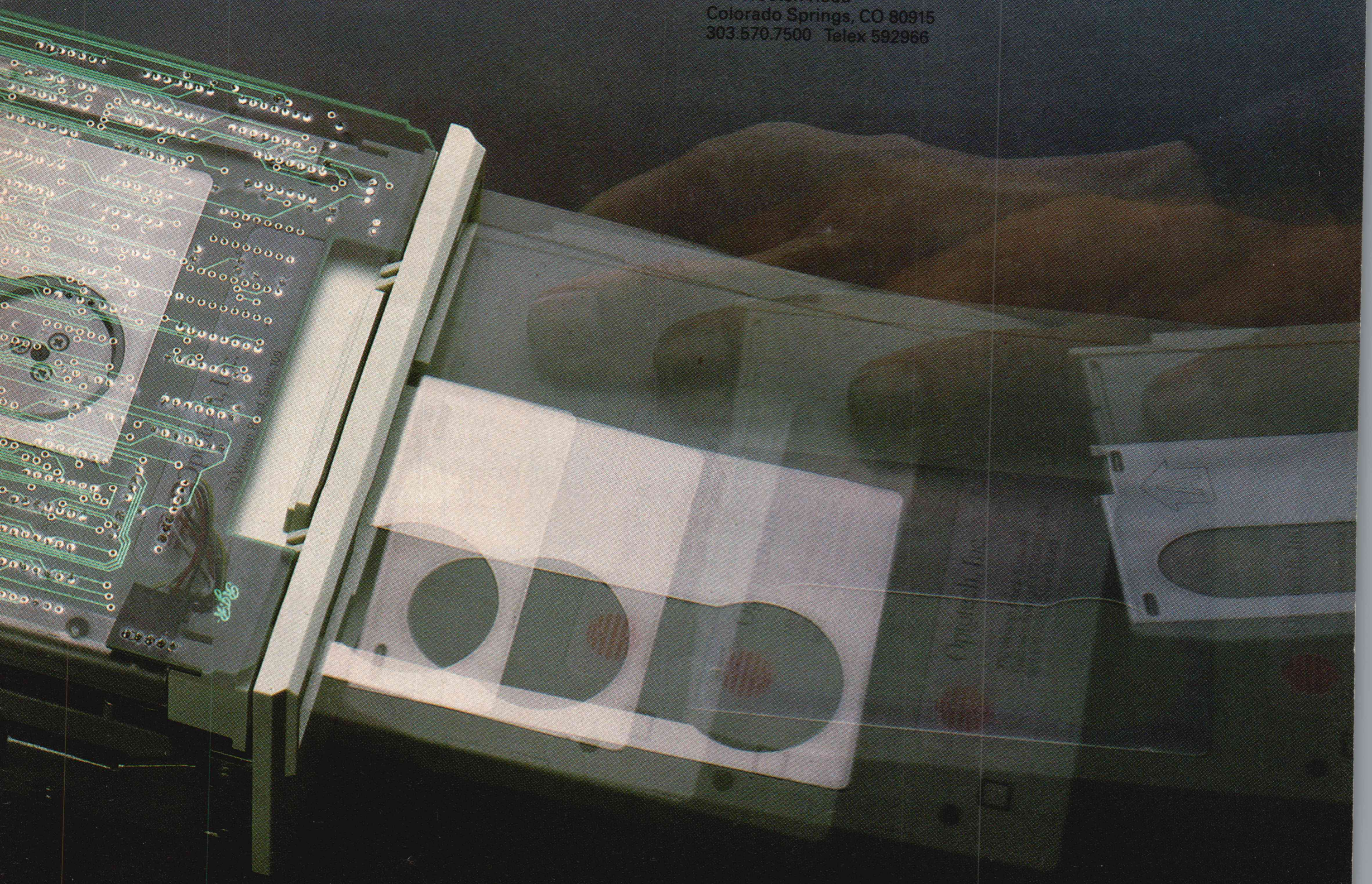
Optical Disk Drive 5984

- *200 megabytes on a removable cartridge.*
- *Fast access read and write.*
- *For PCs and minis.*
- *Extensive interface software.*
- *Available Now.*

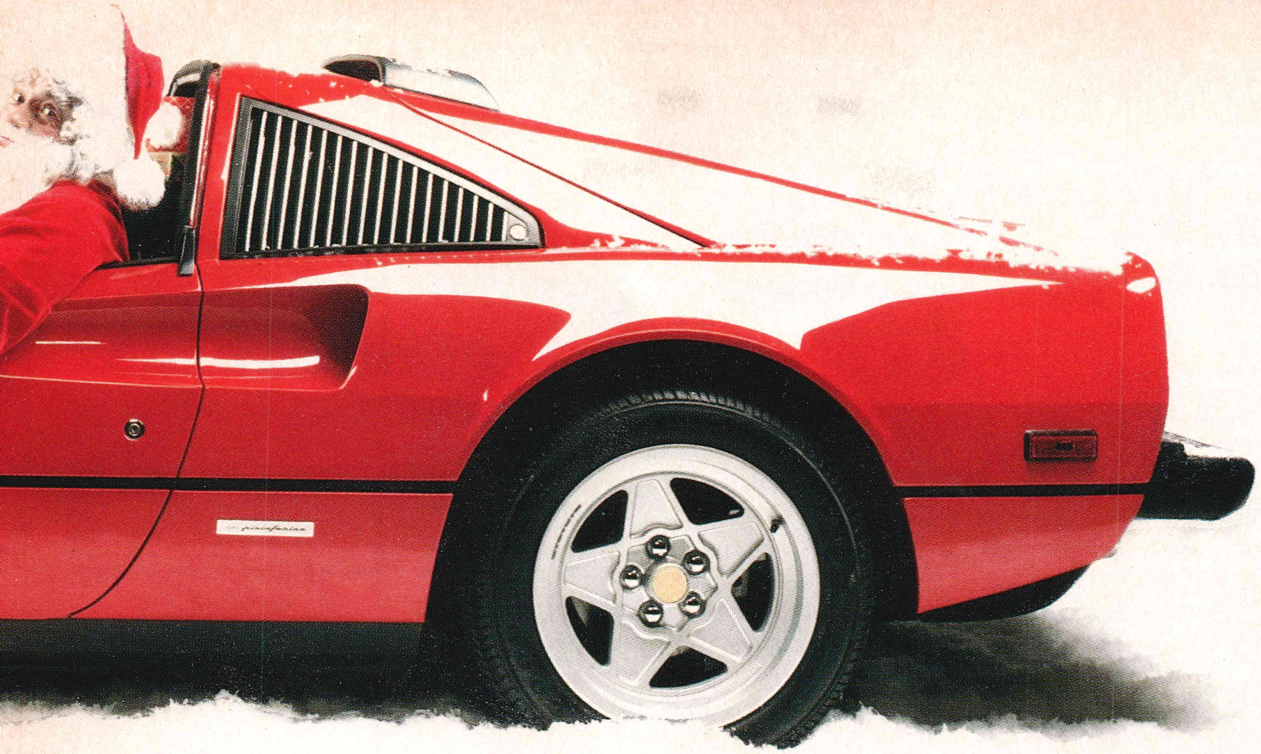
The Preferred Solution For:

- *Database–large, portable, indelible and updatable*
- *Online Mass Storage–integral backup*
- *Imaging–capacity with removability*

Optotech, Inc.
770 Wooten Road
Colorado Springs, CO 80915
303.570.7500 Telex 592966



Circle **no. 124** on reader service card.



y Pac™ For The Holidays.

5. $995.*

16-bit processors is supported, along with the user-assignable string and numeric variables, up to 100-step trace-back, and debug log to disk or printer. Breakpoints can be set in the source file window, whose display can be synchronized with the disassembled object during tracing; symbols can be added incrementally during the debug session; and the disassembly written to disk.

...Pfinish,™ the program that maximizes your program's efficiency...

Pfinish helps you turn your beta-test product into a software work of art. It analyzes your program or the entire operating environment during execution, and produces reports which tell you how much time was spent in each routine or interrupt, who called it, how many times, and much more. Wasteful, inefficient, or non-optimal areas of code become immediately apparent, whether they're in your code, the compiler library, or in the operating system routines themselves.

...Ptel™ gets the lead out of binary file transfer...

You get error-free file transfer and access to mainframes, minis, and micros. ASCII. XModem. Modem7. Telink and Kermit. MS-DOS® 2.x or 3.x. With Ptel. You can transfer 8-bit binary files over a 7-bit data path with Kermit. Masked file-name transfers with XModem. Or, transfer whole subdirectories with a single command using Telink. Ptel keeps track of the original file size and creation date. Ptel also offers a script language, backward scrolling, and the ability to handle DOS commands from inside the program.

SEE US AT FALL COMDEX

...And, Phoenix's new on-line software update service. Free to all registered Pfantasy Pac1 owners.

Pfantasy Pac1 software works in MS-DOS/PC DOS environments, is completely language-independent, and is available for the IBM® PC, XT,™ AT,™ and compatibles, the Wang® PC, the TI Business-Pro,™ and the Tandy® 2000.

So, spoil yourself a little. Get Pfantasy Pac1, and make your favorite programmer's dream come true.

For more information contact: Phoenix Computer Products Corporation, 320 Norwood Park South, Norwood, MA 02062. Or call: (800) 344-7200. In Massachusetts (617) 762-5030.

Programmers' Pfantasies™
by

Phoenix

Programmers' Pfantasies, Pfantasy Pac, Pmaker, and Pfinish are trademarks of Phoenix Computer Products Corporation. Pmate, Pfix86 Plus, Plink86 Plus, and Ptel are trademarks of Phoenix Software Associates Ltd. MS-DOS is a registered trademark of Microsoft Corporation. Wang is a registered trademark of Wang Laboratories, Inc. Tandy is a registered trademark of Tandy Corporation. UNIX is a trademark of AT&T Bell Laboratories. IBM is a registered trademark, XT and AT are trademarks of International Business Machines Inc. TI Business-Pro is a trademark of Texas Instruments Incorporated.

Dr. Dobb's Journal

Editorial
Editor-in-Chief *Michael Swaine*
Managing Editor *Frank DeRose*
Technical Editor *Alex Ragen*
Editorial Assistant *Sara Noah*
Contributing Editors *Robert Blum, Dave Cortesi, Ray Duncan, Allen Holub*
Copy Editor *Vince Leone*
Typesetter *Jean Aring*

Production
Production Manager *Bob Wynne*
Art Director *Shelley Rae Doeden*
Production Assistant *Alida Hinton*
Cover Artist *Kazuo Morita*

Circulation
Sub. Fulfillment Mgr. *Stephanie Barber*
Subscription Mgr. *Maureen Kaminski*
Book Marketing Mgr. *Jane Sharninghouse*
Single Copy Sales Mgr. *Stephanie Barber*

Administration
Finance Manager *Sandra Dunie*
Business Manager *Betty Trickett*
Accounts Payable Supv. *Mayda Lopez-Quintana*
Accounts Payable Asst. *Denise Giannini*
Billing Coordinator *Laura Di Lazzaro*
Accountant *Marilyn Brown*
Adm. Coordinator *Kobi Morgan*

Advertising
Advertising Director
Shawn Horst *(415) 424-0600*
Systems Manager *Ron Copeland*
Administrative Manager *Anna Kittleson*
Advertising Sales
Walter Andrzejewski *(617) 567-8361*
Lisa Boudreau *(415) 424-0600*
Beth Dudas *(714) 643-9439*
Michele Beaty *(317) 875-0557*

M&T Publishing, Inc.

Chairman of the Board *Otmar Weber*
Director *C.F. von Quadt*
President *Laird Foshay*

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0278-6508**

Customer Service: For subscription problems call: outside CA 800-321-3333; within CA 619-485-9623 or 566-6947. For book, back issue, or disk order problems call 415-424-1474.

Subscription Rates: $25 per year within the United States, $46 for airmail to Canada, $62 for airmail to other countries. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank.

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 686-1520 TELEX: 620430 (WUI)

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.

November 1985
Volume 10, Issue 11

CONTENTS

In this Issue

Mark Edwards's review of ten programmable editors is the centerpiece of this issue, and we think it is another step in the direction we want to take in reviewing technical software tools. Edwards has asked all the questions we would ask about a programmer's editor and has tabulated the results; and he's done it for ten editors.

This month we also focus on Modula-2, the language that Pascal's father Niklaus Wirth thinks will replace Pascal for serious programming. Brian Anderson gives a little lesson in Modula-2 programming while presenting his bit-manipulation routines, and David Carroll's comparison of Modula-2 and Pascal introduces the language to those who have not yet encountered it. We've also supplied a brief list of sources of Modula-2 compilers and tools.

This issue's cover image, courtesy of Aurora Systems, is a work of computer art by Kazuo Morita of Tokyo.

Next month we present our second annual operating systems issue. We expect to offer a variety of tools to extend operating system performance for a variety of operating systems. We'll also examine three "operating environments" for 8088-type machines: DRI's GEM, IBM's TopView, and Microsoft's Windows, looking at how they work with MSDOS and at some of the subtler programming issues they raise.

Dr. Dobb's Journal

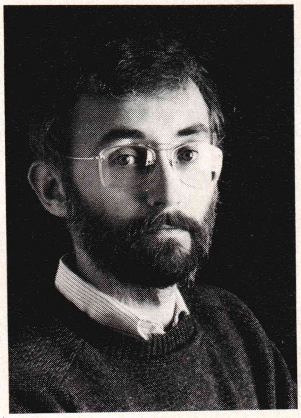
ARTICLES

REVIEW

COLUMNS

DEPARTMENTS

EDITORIAL



In late August and early September, with no trace of embarrassment, *PC Week* announced the "first tangible results of AI research;" *Electronics* described new products that "move AI closer to the computing mainstream;" and *Computerworld* proclaimed that "AI is on the road to reality."

These claims that AI is once more just around the corner were prompted by product announcements at the International Joint Conference on Artificial Intelligence in August and expectations of product announcements at Comdex this month. We've heard such claims before, but this time they may be accurate.

It's not just the fact that Lisp machines and AI workstations are getting cheaper or the fact that Lotus, Ashton-Tate, and Microsoft are ostentatiously spending money on AI development. There are real products coming to market that employ real AI features.

Two recently announced database programs, ANSA Software's Paradox and Symantek's Q&A, employ AI techniques, as does Javelin Software's Javelin financial-modeling program. Q&A's natural-language interface is a descendant of Symantek founder Gary Hendrix's Ladder system that he began developing while at SRI in the 1970s. (These developments and the Borland-Analytica merger, with its prospect of a powerful $99 database package, should at least make the database field more interesting.)

Hendrix's work was originally done in Lisp and, although a germ of Lisp survives in the product, major portions are written in C and assembly language. This is a direction that other AI programs are taking as well. Teknowledge is porting its S.1 knowledge-based system-development software from Lisp to C for memory efficiency, performance, and—most notable—for compatibility with what they refer to as "conventional computing practice." "Conventional computing practice," Teknowledge is telling us, means C.

Neither Teknowledge nor any other firm thinks C is a good language for developing AI applications, but it's obvious they regard it as the appropriate language for delivering systems that require programmer involvement. Lisp is still the leading language in AI development—a status that won't be hurt by the current shift toward Common Lisp (which the trades will inevitably and casually label the standard, de facto or otherwise). HP, Apollo Computer, Intellicorp, DEC, and Xerox have all announced Common Lisp implementations or products based on Common Lisp. Prolog, the other significant AI programming language (which was written explicitly for AI work) gained support when IBM introduced a Prolog implementation for its VM operating system. But, as Michael Genesereth and Matthew Ginsberg point out in September's *Communications of the ACM*, the resolution principle is probably not an adequate deductive method for really effective logic programs, and Prolog is little more than a software implementation of the resolution principle. It's possible that we don't yet have the AI language we need to develop powerful AI-based commercial software.

Then again, IBM's new expert-system development product was written in Pascal—a reminder that you don't necessarily need just the right tool for the job.

At any rate, we at *DDJ* are optimistic enough about the near-term applicability of AI techniques that we're publishing another issue with this theme in April. But a good application of AI techniques is appropriate to any issue, so consider yourself invited to send us yours.

Michael Swaine

Michael Swaine

New Version

Sizzling C.

The fastest C. The C that Microsoft developed to write its own software programs. Hot.

So hot that we can make this claim: Virtually every program runs faster with Microsoft® C Compiler than with any other MS-DOS® C compiler.

Efficient C.

We give you everything you need to write code so tight your computer will scream.

"Preliminary testing on the Microsoft C Compiler produced code that was significantly smaller than that produced by other C compilers."
Paul Springer, Ashton-Tate.

"We found the FAR pointer very helpful for situations where a mix of memory models offers the greatest efficiency."
Robert Frankston, Software Arts.

"The portability of the code between MS-DOS and XENIX® is great."
Jim Bean, Peachtree Software.

But it's really no surprise that our C stretches your micro to its limits. We wrote both the MS-DOS and the XENIX operating systems.

For the name of your nearest Microsoft dealer, or to upgrade from Microsoft C Compiler or Lattice C, call (800) 426-9400. In Washington State, Alaska, Hawaii and Canada, call (206) 828-8088.

And make your programs really cook.

MICROSOFT®
The High Performance Software™

Microsoft C Compiler Version 3.0

Microsoft C Compiler
- Produces compact code and fast executables.
- Implements register variables.
- Small, Medium and Large Memory model Libraries-Mix models with NEAR and FAR pointers.
- Transport source and object code between MS-DOS & XENIX operating systems.
- Library routines implement most of UNIX System V C library.
- Choose from three Math libraries and generate in-line 8087/287 instructions or floating point calls.
  - Floating point emulator (utilizes the 8087/287 if installed).
  - 8087/287 coprocessor support.
  - Alternate math package—extra speed on systems without an 8087/287.
- Link routines written in Microsoft FORTRAN (V 3.3 or higher), Microsoft Pascal (V 3.3 or higher) or Microsoft Macro Assembler.
- Supports MS-DOS pathnames and Input/Output redirection.
- File sharing and record and file locking is supported.
- Do source level debugging, with the Symbolic Debug Utility, available separately with the Microsoft Macro Assembler Package.*

Library Manager
Create, organize and maintain your object module libraries created with Microsoft languages.

Object Code Linker
- Simple overlay linker combines relocatable object modules created using Microsoft Languages into a single program.
- Link very large programs (over IMB, using overlays).

EXEPACK Utility
A new utility to compress sequences of identical characters from an executable file and to optimize the relocation table.

EXEMOD Utility
A new utility used to modify the fields in the header according to the instructions given by the user in the command line.

'C' Benchmarks—done on a Compaq Plus with 512k memory with no 8087. Program "SIEVE," with register variables.

	Exec Time	Code Size	EXE Size
Microsoft C	:9.39	141	5,914
Lattice C	:12.24	164	20,072

*Purchase both Microsoft C Compiler and Microsoft Macro Assembler and get a $25 rebate direct from Microsoft. See package for details.

Replies to "Turbo Pascal vs. the Standard"

Dear *DDJ*,

We are responding to "Dr. Dobb's Clinic" on TURBO PASCAL (July 1985). Hopefully, the following will clarify some of the statements made by Mr. Cortesi in the article.

- The comments about TURBO PASCAL running under CP/M 2.2 and CP/M Plus are not accurate. Please refer to your own review of TURBO PASCAL in your June 1984 issue (referenced below).
- The discussion about the way the compiler treats available memory storage needs clarification. The programmer has total control over the amount of storage allocated by a TURBO PASCAL program. A brief glance at the CP/M-80 memory map (on page 291 of the Version 3.0 Reference Manual) reveals why the amount of available memory must be fixed at compilation time: static variables start at the end address and stretch downwards. For this reason, both Version 2.0 and 3.0 allow you to lower the end address of your program and thus enable .COM files to work on systems with less memory available.
- Regarding TURBO PASCAL and educational institutions, our compiler is well received and widely used. Computer science departments at more than 400 universities around the world have chosen TURBO PASCAL to be their "standard" programming language; ACM has selected TURBO PASCAL as the language for its national and international programming contests; and the Advanced Student Placement Testing Service has designated TURBO PASCAL its official testing language.

This brings us to the main thrust of Mr. Cortesi's article. He points out the ways in which TURBO PASCAL is nonstandard and states that "Borland has an obligation" to support all standard Pascal features. We give careful consideration to our users' requests for enhancements and additions to TURBO PASCAL and, unlike most software companies, we have already implemented many of these features in our two major upgrades. We will consider Mr. Cortesi's suggestions within this context.

Why do we use the Jensen/Wirth report? We consider it the most important standard and, for the most part, follow its guidelines. We are not the only ones to downplay the differences between ISO and Jensen/Wirth. *Dr. Dobb's* says, "For most users, the difference between the two standards will be small. . . . The compiler [TURBO PASCAL] accepts almost all standard Pascal statements. It is therefore possible to write highly portable programs" (June, 1984). We have carefully designed efficiency into our compilers—there are plenty of large, cumbersome systems available that provide total standard support.

TURBO PASCAL provides all the tools and utilities necessary for the vast majority of programmers to produce fast, high-quality scientific, business and personal software. This speed and power have made TURBO PASCAL the industry standard and the choice for more than 400,000 programmers. TURBO PASCAL's price, documentation and support have made it the best software value in the business.

BORLAND INTERNATIONAL
4113 Scotts Valley Dr.
Scotts Valley, CA 95066

I overstated the problems with CP/M storage allocation—see Winograd's letter below. Regarding all other points, I stand by my prior statements.—D.E.C.

Dear *DDJ*,

I read with interest D. E. Cortesi's comments about TURBO PASCAL in the July "Dr. Dobb's Clinic." He is correct in noting that because TURBO PASCAL doesn't look at the BDOS entry address at location 6, a program compiled with a larger TPA will not run in an environment with a smaller one. I originally experienced the same problem. Text processing programs that I wrote in TURBO PASCAL wouldn't run from the WordStar no-file menu, and programs I compiled on my Kaypro 10 crashed on my Otrona, which has a smaller TPA.

Fortunately, there is a very simple solution to the problem. When you command TURBO PASCAL to compile to disk (to do this you use the O and C options from the main menu), you will see a screen such as is found in the Table, page 12. This screen reflects the memory arrangement of my Kaypro 10.

The Start Address is the address for the first byte of the program code. This is one byte higher than the end of the TURBO run-time library. The End Address is the highest address for program code, which is normally (BDOS − 1), minus 700 to 1,000 bytes for the loader which allows other programs to be executed from the TURBO main menu (see p. 261 of the TURBO PASCAL 3.0 manual).

To solve the problem raised by Mr. Cortesi, simply reset the End Address from the above screen. Start by pressing E. TURBO PASCAL will then prompt

End address: A100

The Ultimate Programmer's Editor
WENDIN'S XTC™

VERSION 3.0 STILL ONLY $99^{00}

SUPER PROGRAMMERS edit in XTC to make software development a snap! Just look at these powerful features:

MULTITASKING

XTC's built-in multitasking lets you run your macros in the foreground or independently in the background while you continue editing. A background process has full access to editor resources, and can be used to translate code from one language to another in REAL TIME, print files in the background, or even scan syntax while you type in code. Best of all, you can use XTC to edit source and documentation in any programming language!

COMPILE IN WINDOWS

All DOS compilers and utilities can be executed from within XTC using a single keystroke. While it runs, XTC captures your compiler's output and redirects it into your text, so you can compare compiler messages with your source code ON THE SAME SCREEN. And using XTC's macro language, Turbo Pascal is literally only a keystroke away. You can use other compilers and utilities inside XTC too — like Lattice "C," Microsoft Pascal, and IBM's Basic, to name a few.

MACRO LANGUAGE

XTC has the most powerful macro language in the editing world. XTC's macros aren't just keystrokes assigned to keys; they're real programs that can be used to automatically edit source code and data files. Like any real programming language, XTC has control structures like IF THEN ELSE, WHILE DO, REPEAT UNTIL, FOR NEXT, DUPLICATE N TIMES, INDEFINITE LOOP, EXIT, and BREAK LOOP. XTC also has INTEGER, BOOLEAN, and STRING variables to hold numbers, conditions, and pieces of text.

WINDOWS & BUFFERS

With XTC you can display up to 8 different files or parts of the same file on the screen at once. XTC's windows are programmable and can even be linked together to share files. XTC also has 20 other buffers that you can use to hold files and blocks of text.

WORDSTAR COMPATIBILITY

If you already know Wordstar commands, then you don't need to learn a new set of commands. If you want to customize XTC, just write macros to emulate the key layout you're used to. XTC can also read Wordstar files, and can even strip off all of the non-standard high bits with a single command.

LARGE FILE EDITING

XTC lets you edit files entirely in memory (using all available memory), or paged to disk, for maximum flexibility. You can choose how XTC buffers text.

INCREDIBLE EXTRAS!

- UNDO N TIMES
- REMOVE WORDSTAR HIGH BITS
- EDIT GRAPHICS DISPLAYS
- AUTOINDENTING MODE
- TAB EXPANSION/ COMPRESSION MODE
- EXTRA LONG LINES
- MACRO COMPILER
- TELEPHONE SUPPORT
- 150 PAGES OF DOCUMENTATION
- RUNS ON IBM / PC, XT, AND / AT COMPUTERS (AND TRUE COMPATIBLES)

COMPLETE SOURCE CODE

XTC comes with 7,000 lines of source code jam-packed onto two DSDD disks. Includes 13 modules written in Pascal, and 2 assembly libraries you can use to access the PC's screen, intercept software interrupts (like INT 21H functions), allocate and deallocate memory, and load and execute programs. It's all included FREE for your recreation and enjoyment!

INCLUDED FREE!
Macro Compiler with full source code in "C"! Translates Ada control statements into XTC macros. It even runs in XTC windows!

ORDER HOTLINE
509/235-8088
CREDIT CARDS WELCOME!

XTC outperforms any other programmable editor on all IBM/PC, /XT, and /AT computers (and true compatibles). If you want to feel the power of XTC before you buy it, just ask for our demo disk (only $10) and try it out. When you buy XTC, we'll knock 10 bucks off the price.

To get your copy of XTC now, order it over the phone — we can ship it the same day you call! Or, send in an order, just like this one:

XTC	**$99.00**
Macro Compiler	**FREE**
Shipping, Handling, Insurance	**3.50**
Want it COD? Add this	**1.90**

TOTAL IT UP, AND SEND IT QUICK!

WENDIN™

BOX 266
CHENEY, WA 99004

The people who make quality software tools affordable.

Circle **no. 112** on reader service card.

As shown by the entry above, I usually choose A100h as my End Address. This ensures that a program will execute on any system with a 40K TPA. This should be enough even for ancient homebrew systems, and it should allow programs to run on almost any machine from the WordStar no-file menu or with memory-resident utilities, such as keyboard enhancers, installed. If not, simply go back to the above menu and increase the End Address until the "Not enough memory" message stops appearing when the compiled program is run.

Ed Winograd
4704 Edison Lane
Boulder, CO 80301

All true and I should have known it. But this is still very restrictive: you have to size your program to the worst possible case, and it can never take advantage of storage over that minimum.—D.E.C.

Dear *DDJ*,
While it is apparent that Mr. Cortesi possesses a working knowledge of Pascal, it is also apparent that he has not worked with standards organizations. He makes such statements as "the standard" and "There is a Pascal standard," giving one the impression that there exists "one and only one" standard for Pascal. There could be nothing farther from the truth! Secondly, having been personally involved with several projects keyed against emerging standards, "draft," "final draft," and "standard" do *not* have the same meaning, practically or in any other sense of the word!

Mr. Cortesi then proceeds to tell me that ANSI, IEEE, and ISO are one big happy family. First, he ignores the fact that each one of these independent standards organizations has published their "standard" for Pascal separately, identified by each organization's individual "numbering/labeling" plan. Mr. Cortesi should have identified each standard by appropriate label/name, organization, and date, and then identified which of these was "the standard" for the purposes of his article.

Finally, I find it particularly difficult to accept the premise that merely because a programming language is standard (adheres to some recognized body's standards) it is in some way inherently portable. I would be extremely pleased to see your journal tackle this very volatile issue in the near future.

Bruce Hutfless
Tacoma, WA

There is just one U.S. domestic Pascal standard. It was the joint work of the IEEE and ANSI and was published in identical form by both bodies under the title American National Standard Programming Language Pascal. *The foreword to the document details the close liaison that was maintained between the U.S. joint committee and its international counterparts. There is a formal statement: "Differences of Technical Substance Between the Standard and the International Standard, as Represented by ISO/DIS 7185." Except for the U.S. omission of conformant arrays, these are exceedingly narrow technical points. There's a careful statement of how "compliance with this standard is equivalent to compliance at level 0 with the international standard." The joint committee's intent is quite clear: they mean for the U.S. and international standards to agree with each other. For all practical purposes, there is true worldwide agreement on a single standard for Pascal.—D.E.C.*

Dear *DDJ*,
Allow me to congratulate you on your excellent critique of TURBO PASCAL that appeared in the July issue of *DDJ*. Frankly, I wish that someone else had written it, since many people still remember Dave Cortesi's review of JRT Pascal and his reputation of being picky about Pascal implementations may lessen the impact. Nevertheless, he certainly said things that desperately needed to be said. All the uniformly glowing praise for TURBO was beginning to bother me. Finally someone has realized (in print) that, as implementations of Pascal go, TURBO is very mediocre.

My own experience with TURBO PASCAL has been rather unpleasant. Over the past three years I have written roughly 60,000 lines of Pascal code for a wide variety of applications. I do almost all my work under VAX/VMS, which has an absolutely first-rate Pascal compiler (full support of the ISO standard with optional warnings on nonstandard features, highly optimizing, 2000 lines/minute average compile time on a VAX-11/750, source-level debugging, etc.).

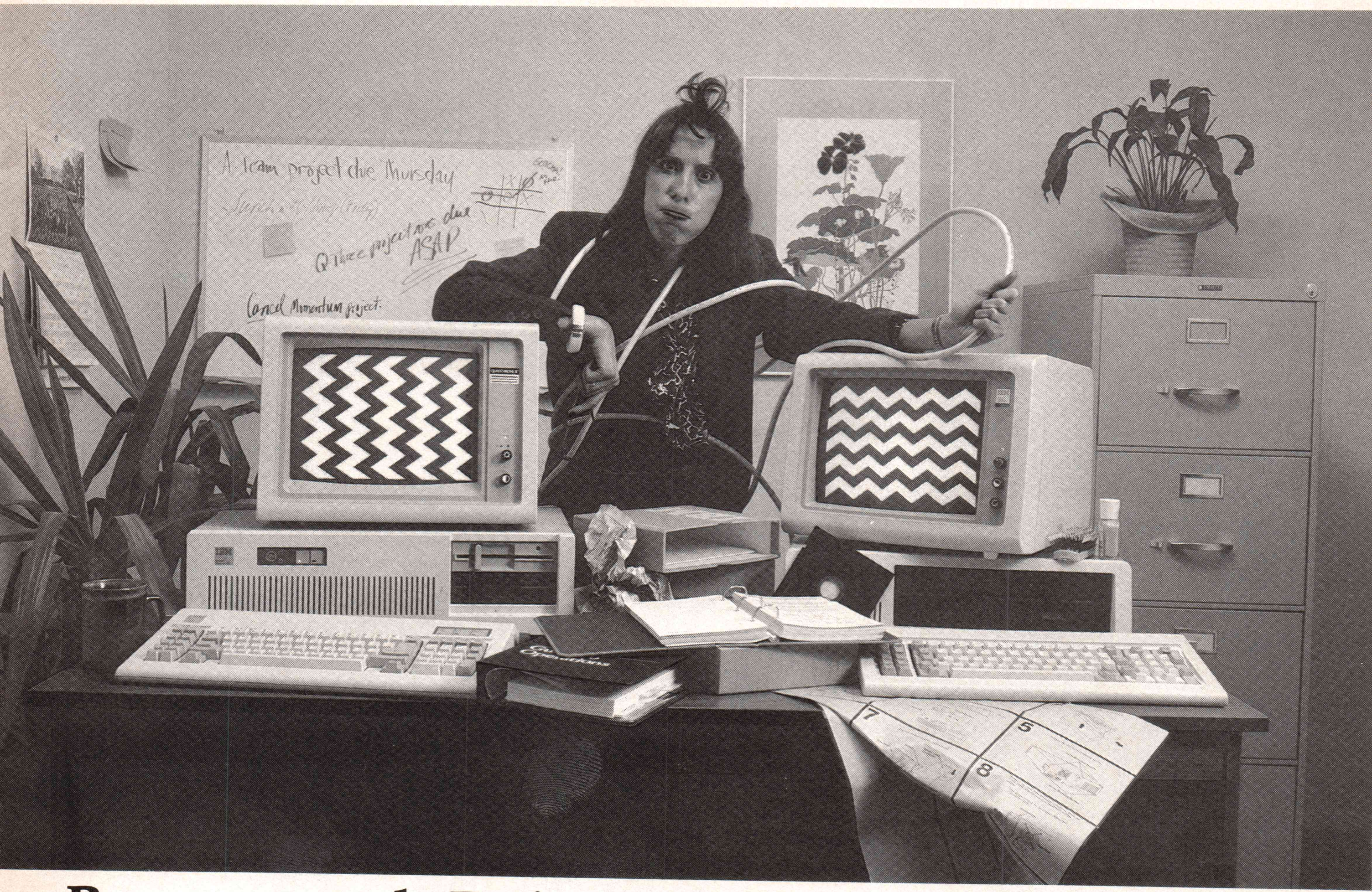
I get a request to port code to MSDOS about once a week. When TURBO first became available, it sounded great. The reviews made it sound like the ultimate compiler. I borrowed a manual from a friend and read it. It was obviously Pascal implemented by people who did not understand the basic philosophy of the language. Worse, the implementation was completely unusable. I write strictly standard-conforming code most of the time, but I frequently use all the features TURBO elected to omit, many of which are virtually impossible to work around. The result is that TURBO is absolutely useless. This is very difficult to explain to neophyte TURBO owners who want to use my code. "Buy a real Pascal compiler instead of a compiler for some random language invented by Borland" does not tend to go over well.

After countless arguments about standards (and more trips to the bookshelf) my customary response now is to simply give people the code they want and tell them to go for it. To forestall the inevitable arguments I also give them a listing with all nonstandard features flagged by the VAX Pascal compiler. After a few attempts they usually give up or just switch to some other implementation.

Edwin Earl Freed
Box 430
Perkins, OK 74059

Dear *DDJ*,
I read your Clinic column in the July 1985 issue with particular interest because I teach an introductory programming course to engineering students using TURBO PASCAL. Your points have generally been well

If you can't share files on PC Network, you're using the wrong file manager.



Be connected. Btrieve.™

Networks can solve problems. But running a single-user file manager can create new ones: Lost updates. Garbled data. Trashed files.

Btrieve™/N offers *safe* multi-user file management that protects your data when sharing files. And eliminates the need to rewrite your application for networking. Btrieve/N set the file management standard for the industry's most popular LANs: Netware, Omninet, PC Net, EtherSeries, Nestar, and NetOne. And now Btrieve/N sets the standard for IBM's PC Network.

Fast. Btrieve/N is fast, too. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing with automatic balancing for access speed that won't degrade as your database grows. With Btrieve/N, your applications always run fast. And users don't waste time waiting.

Automatic file recovery. Btrieve/N provides automatic file recovery after a system crash, so accidents and power failures don't turn into database disasters. Your Btrieve data always comes back intact.

Fully-relational data management. Using SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™/N adds report writing capabilities for generating the reports you need. Xtrieve™/N speeds users through database queries with interactive, on-screen menus—no command language or special syntax.

For professional programmers. Btrieve/N is the fast, reliable answer for all your application development. In any development language—BASIC, Pascal, COBOL, C, FORTRAN, and APL. With multikey access to records. Unlimited records per file. Duplicate, modifiable, and segmented keys.

With Btrieve/N, you can develop better network applications. And solve problems, not create new ones.

Now available for XENIX

Suggested retail prices: Btrieve, $245; Btrieve/N, $595; Xtrieve, $195; Xtrieve/N, $395; Rtrieve, $85; Rtrieve/N, $175. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X.

Btrieve, Xtrieve, and Rtrieve are trademarks of SoftCraft Inc.;

SoftCraft Inc.
P. O. Box 9802 #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

Circle **no. 118** on reader service card.

made, and I can only hope that Borland does the necessary to improve compatibility. Although Version 1.0 undoubtedly was written on a shoestring and in a hurry, there has been plenty of time (and money) for improvements to Version 3.0. In fact, I expect to continue using Version 2 because there has been so little improvement in Version 3.0.

I do expect to continue to use it for teaching, and I don't agree with your objection to its use for this purpose. Because of the fast compilation and built-in editor, students learn faster than with any alternative I have seen. What they learn appears to be valid "standard Pascal". You have not identified any features that are violations. There is more than enough to keep student programmers busy for a semester (or two) without ever getting to the New/Dispose and Get/Put problems, although the omission of the Page procedure is disappointing. They should be able to progress to another, more complete compiler if they really need such advanced features.

Edward H. Wiser
North Carolina State University
School of Agriculture and Life Sciences

Dear Mr. Kahn,
I recently purchased a number of copies of the MSDOS implementation of Version 3.0 of TURBO PASCAL for our department. Although there is much to be pleased with, I have a number of concerns that are very substantial. Fortunately, as I was gathering my thoughts on the subject, there appeared in the July issue of *Dr. Dobb's Journal* an article, by D. E. Cortesi, which expresses my views to perfection. I assume you will have seen it. Personally, I find the absence of Get and Put an enormous handicap. Their replacements are really pernicious.

To Mr. Cortesi's implied requests for a version of TURBO PASCAL that implements Pascal in a more "standard" way, let me add my own. The current implementation will not be acceptable for use in classes that teach Pascal. I will advise those I know to avoid using it.

Phillip Emig
Department of Mathematics
California State University, Northridge
Northridge, CA 91330

Dear *DDJ*,
I sent you a letter last month in which I demonstrated the way to correct a problem in TURBO PASCAL's code structuring. However, it seems that the letter must have been lost in the shuffle somewhere, as D. E. Cortesi's column in the July issue claims that TURBO PASCAL is "... unusable under CP/M Plus, and dubious under CP/M 2.2". The problem he describes is real, but as can be seen in my letter, the fix is extremely simple. And to date, no one else has offered or published this procedure.

Sincerely,
James R. Shiflett
P.O. Box 1236
Stafford, TX 77477

Mr. Shiflett had sent a discussion of how the CP/M version of TURBO allocates storage, including an involved method by which you can trick it into using all available storage. He's put the same information up on the Borland SIG on CompuServe. I thought that, while he'd done a first class job of systems work, his solution was a typical expert's hack: unsafe, prone to error, dependent on a particular compiler version and a particular operating system and thus absolutely nonportable, and in general not something I would want to use or publicize.—D.E.C.

TEX Features

Dear *DDJ*,
Thank you for the fine review you gave our product, PCTEX. We hope you will give Personal TEX a chance to mention recent enhancements to our product. We might add that these are not proposed additions, but actual features that are part of the product being shipped today.

- We have, now, a TEX driver for the Corona Laser Printer. It produces 300-dot-per-inch TEX finals at two 8½-by-11 pages per minute on an XT and six pages per minute on an AT.
- We offer a complete TEX publishing package for the PC consisting of PCTEX, the Corona driver, and a Corona Laser Printer, all for only $3,395.
- We carry the complete line of Textset device drivers (Apple LaserWriter, screen preview, and more) and OCLC Metafoundry fonts (Helvetica, Copperplate, Schoolbook, and others). (Lack of screen preview was one of the complaints of the reviewers.)

Our unbundled basic TEX package is still only $279—a little over half of the competition's bundled price. And we've got Michael Spivak, the gadfly author of *The Joy of TEX*, on our team. Michael has written a delightful introduction to TEX together with an easy-to-use macro package he calls VANILLA that comes only with PCTEX.

We would like to let potential buyers of document-formatting software know that TEX is all we do. When you talk TEX with us, you've got 100% of our attention.

By the way, we will soon be shipping our implementation of TEX, Version 1.5.

Lance Carnes
President, Personal TEX, Inc.
20 Sunnyside Avenue
Suite H
Mill Valley, CA 94941

We have received an announcement of an agreement under which Addison-Wesley, distributors of MicroTEX, will also license and distribute the Textset laser printer and screen preview drivers.—Ed.

```
                 Memory
compile →        Com-file
                 cHn-file

Start address:     20E2 (min 20E2)
End address:       CA42 (max CD06)

Find run-time error Quit

>
```

Table
Adjusting TURBO Memory

DDJ

NEW !! NEW !!

TURBO
INTEGRATED DEVELOPMENT SYSTEM

These systems were developed from the ground up to work in an integrated fashion ... This saves you time and money from trying to assemble different packages and to code in our Products Unique Features.

TURBO ISAM MASTER

GENERATES 'Ready to Run' Turbo Programs Using BORLAND'S Turbo Pascals Database TOOLBOX.

With a few key strokes you can generate the following Pascal programs/includes files. Automatically interfaces to Turbo Screen Master.

A. MASTER DATABASE PROGRAM - Generates Pascal Program Code for the following functions.

1. Add a Record ● Allows both duplicate & unique only keys. ● Change the up to 6 fields comprising the primary key and the up to 3 secondary keys at any time during input, and the validity of the keys are checked and the ISAM key files are automatically adjusted.

2. Delete a Record - Shows the record to be deleted on the screen and allows you to change your mind about deleting, and then adjusts the keys files automatically for the up to 4 keys. The disk space is reused automatically by the programs.

3. Edit a Record - with a key change allowed.

4. Search Database by Key - Up to 4 keys-and has a 1 key lookup to find next key in alphabetic order.

5. For Each Key - The ability to display the keys on the screen or printer in sorted order.

B. DATA BASE RECOVERY PROGRAM. This program recovers your data base if it is corrupted by a power outage or certain hardware failures.

C. MULTIPLE ISAM FILES - -Can be used at the same time in a single program. The generator also produces context sensitive instructions on how to integrate the generated program into your main program.

D. DOCUMENTATION - Print screens and ISAM specifications. Also inline program documentation is generated.

TURBO MENU MASTER

GENERATES 'Ready to Run' Turbo Programs with Dedicated Screens or Windows.

Features: Has Menu Data Base so its easy to modify menues.
Provides Selection by - **(1) Press of a number**-If you have more than 9 selections then you enter only a single number if the selection is 2-9. **(2) Press of a function key** - If more than 10 selec tions then can enter use shift function key to get up to 18 selections. **(3) Press the High Lighted Letter** -in the selection description, you choose a letter for each menu selection in the interactive menu generator. **(4) Use the arrow keys** - to point to a menu item & then press return.

Adaptive Screen Colors -Different screen colors are automatically used for color or monochrome display. This allows you to provide a better interface for the user.

Interactive Menu Builder - (1) Allows for the automatic entry & reorder of selections. (2) Offers easy color selection. (3) Allows for choice of procedure, chain, execute or comment code generation for each selection. (4) Has startup Menu that YOU design. (5) Provides for the integrated Display & Control of the Key Lock Status.

RISK FREE TRIAL

Try the demo package included for 30 days. If not pleased return for a full refund.

NO ROYALTIES on Generated Programs

Credit card orders call:

1-800-821-9503

In Florida (800) 342-0137

HAWAIIAN VILLAGE COMPUTER SOFTWARE
1109 Pennsylvania Ave. St. Cloud, Florida 32769

Circle no. 61 on reader service card.

TURBO SCREEN MASTER

Some of the **UNIQUE** features of the **Turbo Screen Generator** are:

Intelligent Cursor - Computer automatically determines which single/double character to use from the Extended IBM Character set to draw boxes and borders. You just move the cursor where you want the border. The other Screen Systems require that you must tediously choose the correct graphic character.

Extended Variable Names - Allows the use of Structures & Arrays in your variable names and automatically generates the correct declarations and initalization routines. Also you are allowed to declare a field larger than the screen display size.

Provides Three Color Selections Methods - Easy to develop color screen on Monochrome Monitor and to develop Monochrome Screens on the Color Monitor.

Adaptive Screen - Entry colors change automatically depending on what type of screen is in use.

COMPARISON WITH OTHER PRODUCTS	TURBO SCREEN MASTER	TURBO SCREEN VER - 1.10	SCREEN SCULPTURE Ver - 1.01
Border Color Control	YES	NO	NO
You Assign Variable Names No cyptic variable names whose names depend on where they appear on screen	YES	NO	YES
Range and Date Checks	YES	NO	YES
Data Entry Valid Character Set	YES	NO	NO
Data Entry Mask Helpful for Profession Screen input & Validation	YES	NO	YES
Initalize Variables to a starting value	YES	NO	YES
Data Entry Valid String Set	YES	NO	NO
Pascal storage for type of Boolean & Integer	YES	YES	NO
Control Capa/Num Look	YES	NO	NO
Auto-Initalization of Date/Time	YES	NO	NO
User Defined Error & Message Handler	YES	NO	NO
Generated program adapts automatically to IBM Screen-Monitor Type	YES	NO	YES
Handles Function Keys	YES	NO	NO
Help Screen Procedures	YES	YES	NO
Optional ISAM Keys Screens Code Generated automatically	YES	NO	NO
Turbo Toolkit Included	YES	NO	NO
Undo Function	YES	NO	NO
Provides Running Time Display	YES	NO	NO
Integrated Data Dictionary	YES	NO	NO
Auto-start Screen for Screen Generation	YES	NO	NO
PRICE (INCLUDES SHIPPING)	**\$89.95**	**54.95**	**125.00**

FREE TURBO TOOL KIT $44.95 VALUE

WITH ORDER OF A MASTER PRODUCT

YES, Enclosed is ____________

INTRODUCTORY OFFER

☐ Turbo Pascal 3.0	~~\$69.95~~	\$59.95
☐ Turbo Database Tool Box	~~\$54.95~~	\$49.95
☐ Turbo Menu Master	~~\$69.95~~	\$54.95
☐ Turbo Screen Master	~~\$124.95~~	\$89.95
☐ Turbo ISAM Master	~~\$149.95~~	\$124.95

Systems Requires: IBM PC,XT,AT or 100% Compatiable -196K. MS DOS 2.0 or Higher - 80 column screen. Turbo Pascal 3.0 - 2 DD/DS Disk Drives. ISAM Program Generator Requires Turbo Database Toolbox.

*These prices include shipping to all U.S. Cities. All foreign orders add $10 per product ordered

For Other Inquires Call (305) 892-5686

Name: ____________ **Phone:** ____________
Shipping Address: ____________
City: ____________ **State:** ________ **Zip:** ________
VISA or MC #: ____________ **Exp. Date:** ____________

C.O.D.'s and Purchase orders WILL NOT be accepted. Outside USA: make payment by bank draft, payable in US dollars drawn on a US bank.

by D. E. Cortesi

Watt Did Ross Dew?

In June we published a short BASIC program by David Ross and asked "Watt duzzit dew?" Several of you wrote to explain it to us. Some provided improvements; we'll get to those in a bit.

Ross's original program appears in Listing One (below). Watt duzzit dew? Richard Oakland of Fon Du Lac, Wisconsin, says, "In theory, Ross's program determines the digits of the number 2^k and places the character representations of those digits into successive elements of a string array of dimension $k+1$. It concludes by printing the array." Why is it so complicated, when

```
130 PRINT 2^K
```

would accomplish about the same thing? "It seems likely," Oakland says, "that the object was to display with full precision numbers that have more digits than single precision can generally handle . . . the author might even have hoped to display numbers that overflow single precision, numbers like 2^{400}, for example."

Why $k+1$ digits? David Barker of Mt. Clemens, Michigan, says that this number "can be justified because it requires $k+1$ digits to represent 10^k, and, because 2^k is less than 10^k, you will always be able to express 2^k in $k+1$ digits or less."

How did Ross want his program to work? Let's go through it line by line, as several readers did. Look first at lines 110 and 140, which are closely related:

```
110 DEF FNL(X)=LOG(X)/T
140T=LOG(10)
```

We don't all work with logarithms every day, so here's a brief review. The BASIC function LOG( ) returns the *natural log* of a number, the logarithm to the base *e*. This function is usually written ln( ) to distinguish it from the *common log*, or logarithm to the base 10, that we met first in school. Why does BASIC provide only natural log? Because it can be used to obtain a logarithm to any base: the log of a number x to any base b is

$$\ln(x)/\ln(b)$$

Function FNL(X) is taking advantage of that identity to provide the common log of its argument X. It would have been clearer if written

```
DEF FNL(X)=LOG(X)/ LOG(10)
```

although slightly slower in execution.

Line 120 gets the power to which 2 is to be raised. It should be forced to be an integer because the program is clearly meant to produce an integer output. You should also test to see whether it is positive; 2^0 is just 1 and a negative power will cause errors later.

Line 130 creates the array of digits and clears it to zeros. Clearing is necessary, says David Shochat of Los Angeles, because "there is one sense in which this algorithm is efficient. The digit calculated at each iteration is actually the next *nonzero* digit. If we use K=10 the '0' in '1024' will be skipped over. This is why the buffer has to be initialized to zeros at the start."

Line 160 sets up the target of the conversion, the number 2^k:

```
160 A0=K*FNL(2)
```

The crux of Ross's program is that this number, which might be very large indeed, can be represented safely in BASIC by its common logarithm. Because of the identity

$$\log(x^k)=k*\log(x)$$

Ross can get the log of 2^k without ever evaluating the number itself.

The Achilles' heel of the program is that, in most editions of Microsoft BASIC, LOG and all the other transcendental functions are evaluated only in single precision, so the value

```
100 REM BASIC PUZZLE BY D. ROSS
105 REM MODIFIED FOR MBASIC BY DEC
110 DEF FNL(X)=LOG(X)/T
120 INPUT "K=",K : DIM A$(K+1)
130 FOR I=1 TO K+1 : A$(I) = "0" : NEXT I
140 T = LOG(10)
150 Z = ASC("0")
160 A0 = K*FNL(2)
170 B = 1
180 WHILE B > 0
190     A = A0+FNL(B)
200     N = INT(A)
210     IF N<0 THEN 280
220     L = A-N
230     X = 10^L
240     D = INT(X)
250     A$(K-N+1) = CHR$(D + Z)
260     B = B-10^(N-A0+FNL(D))
270 WEND
280 FOR I=1 TO K+1 : PRINT A$(I); : NEXT I
```

Listing One

of 2^k is known only to six significant digits, regardless of its size. As Dave Shuman of Pendleton, Indiana puts it, "sizable numbers rapidly exceed the capacity of significant digits calculated in line 160. Even if the rest of the program contained infinite significance, the necessary information has been lost." No doubt, that's why Ross didn't put a DEFDBL at the head of his program. Some BASIC implementations, however, *do* support LOG with greater precision, as we'll see later.

Now that we have some idea of what the program tries to do, let's look at the loop in which it does it. Its first step is

```
190 A=A0+FNL(B)
```

David Shochat's analysis helped us see that here Ross is using another logarithmic identity

$$\log(x \times y) = \log(x) + \log(y)$$

to compute

$$A = \log(B \times 2^k) = \log(2^k) + \log(B)$$

The first time through B=1, so A is just A0. Now the program takes that value apart:

```
200 N=INT(A)
210 IF N<0 THEN GOTO 280
220 L=A-N
```

Shochat says "N and L are the characteristic and mantissa, respectively, of the log of $B \times 2^k$, that is:

$$B \times 2^k = 10^{(N+L)} = 10^N \times 10^L$$

Dave Shuman puts it this way: "Line 200 determines which digit N is next (the integer portion of a logarithm determines the decimal position of its antilog); line 210 terminates the loop in case the logarithm didn't come out exact; and line 220 gets the rest of the logarithm, L, which tells what the digits are beginning at position N."

The next lines process the number further:

```
230 X=10^L
240 D=INT(X)
```

Shochat analyzes this as setting up so that

$$B \times 2^k = X \times 10^N,\ 1 \le X < 10$$

"which is just scientific notation." Shuman seconds him: "the antilog of X is the number in the form a.bcde. . . ." Its integer part, assigned to D, is the next digit to display. The next line,

```
250 A$(K-N+1)=CHR$(D+Z)
```

assigns that digit as an ASCII character to the output array. It isn't hard to see how this code might pick off the *initial* digit of A0, but how does it get the rest of them? The next line is the key:

```
260 B=B-10^(N-A0+FNL(D))
```

Shuman says it removes the new digit

by adjusting the factor B: "At each cycle, if you multiply B times A0, the original power of 2, you should get the rightmost N−1 digits of A0."

Shochat tries to explain how. "On the first iteration B=1, so the first D we get is the leading digit in the decimal representation of 2^k. Now the new B value has to lop off somehow the digit just found. It is

$$B-10^{N-A0+\log(D)}$$

which is

$$B-10^{N}\times(1/10^{A0})\times10^{\log(D)}$$

which is

$$B-10^{N}\times(1/2^{k})\times D$$

Now the trick is to think of this multiplied through by 2^k, which gives

$$(B\times2^{k})-(D\times10^{N})$$

Thus the new $B\times2^k$ is just the old $B\times2^k$ with D's contribution removed. At the end of the iteration where D is the last digit, this will bring B down to zero and end the loop." If it doesn't work out even because of round-off errors, the value of A calculated in line 190 will be less than 1, so the characteristic of its log assigned to N will be less than zero and line 210 will bail out.

Why Doesn't It Work?

This is all very ingenious but, as you know if you've tried the program, it doesn't work very well. Using MBASIC on CP/M or IBM BASIC before Version 3, it produces correct answers for only K=1, 2, and 3 (producing 2, 4 and 8). Then it gets progressively less accurate. How come?

There are two problems. Shuman says, "The most immediate problem is the conversion in line 240, where the real number X is converted to the integer D. For the rightmost digit we run into the classic problem of fixed/float equivalence. The next digit might be 4, for instance, but X was calculated as 3.99578 and D will then be 3. In all but the rightmost digit, the number X will be higher than the digit value, reflecting the digits yet to

come, and the conversion is OK.

"I inserted a new line 280. Because we can tell by inspection that all results for K greater than zero should be even numbers, the new line 280 adds 1 to any results that end in an odd digit."

When this fix is made, the program produces correct answers up to the point where the number of digits in 2^k exceeds the precision of the values in A0, A and B, which are in turn limited by the precision of LOG. It can do K=20 in our CP/M system. Jack Fay of Seattle got to K=24 in Applesoft BASIC. Larry Manns of Trappe, Pennsylvania, got right answers to K=35 on a Wang. By using double precision on all variables in Microsoft BASIC 3.0, Shuman got up to K=51 on an IBM PC/AT. Shochat worked it to K=100 on a VAX.

The fundamental constraint is that, although logarithms allow Ross to cope with the *magnitude* of very large numbers, the limited *precision* of float variables keeps the program from displaying their many digits accurately. "My hat is off to David Ross," Shuman says. "In one elegant little program he gives a demonstration of two classic problems in computer number representation: loss of significance and inexact representation of whole numbers."

The program in Listing Two (page 21) shows our attempt to stretch the precision of MBASIC to its outer limits. It squeezes out two more digits to reach K=24 by using double precision and literals for ln(10) and LOG(2). By some fluke it also gets K=30 right, but not the numbers above or below it.

Other Ways

Two readers tried to do Ross one better. The first was Orin Safier. He simply automated what you'd do to calculate a large power of 2 by hand: you'd write down the largest power of 2 you knew or could look up; then you'd sit there doubling it over and over, producing something like the following table

65,536
131,072
262,144

With a little practice you can compute the digits of the next number almost as fast as you can write them. But why not let a program do the doubling?

Our rewrite of Safier's program appears in Listing Three (page 22). We changed the original only for longer variable names and to use a Boolean expression, not an IF, to handle the carry. Safier says the program "will print 2^k with perfect accuracy" but "involves many iterations for high values of k, and is thus abstractly less satisfactory than Ross's." That's a good point. Can you work out an approximate expression for how many iterations Ross's loop will make as a function of the input, K? What about Safier's?

And here's an extra-credit puzzle for APL users. Can you implement Safier's method, replacing the inner loop with non-looping array operations? You'll have to find a way to handle the carries in parallel.

David Barker also set out to better Ross, but used Pascal instead of BASIC. Like Safier, he aimed to beat the limited precision of float numbers by implementing his own ultralong integers, storing each decimal digit as one entry in an array. But where Safier arrived at 2^k by doubling 1, K times, Barker looked deeper. He explained it to us in terms of binary multiplication. We'd prefer to explain it to you in terms of one of the identities of the exponential function, namely that

$$x^y = x^p \times x^q$$

provided that $y=p+q$ (for example, $2^5=2^3\times 2^2$). Next, notice that any odd number y can be decomposed into a p of 1 and a q of the even number one less than y, whereas any even number y can be decomposed into equal p and q simply by breaking it in half.

Those facts, plus the identity that anything to the first power is itself, practically lead us by the nose to a recursive definition of a power function. The case of x^1 is the one whose answer we know immediately. The rules for decomposition supply us with a way to make the magic recursive statement, "If I knew the answer to the simpler cases x^p and x^q, I could compute x^y:"

```
PowerOf(x,y)
   if y = 1 then return x
   if odd(y) then
      let t = PowerOf(x,y-1)
      let t = t * x
   else
      let t = PowerOf(x,y/2)
      let t = t * t
   endif
   return t
```

Actually, the two legs of the "if" can be collapsed into one to make a simpler function:

```
PowerOf(x,y)
   if y = 1 then return x
   let t := PowerOf(x,y div 2)
   let t := t * t
   if odd(y) then
      let t := t * x
   return t
```

That's a general power function. When we know that x is 2, it gets simpler:

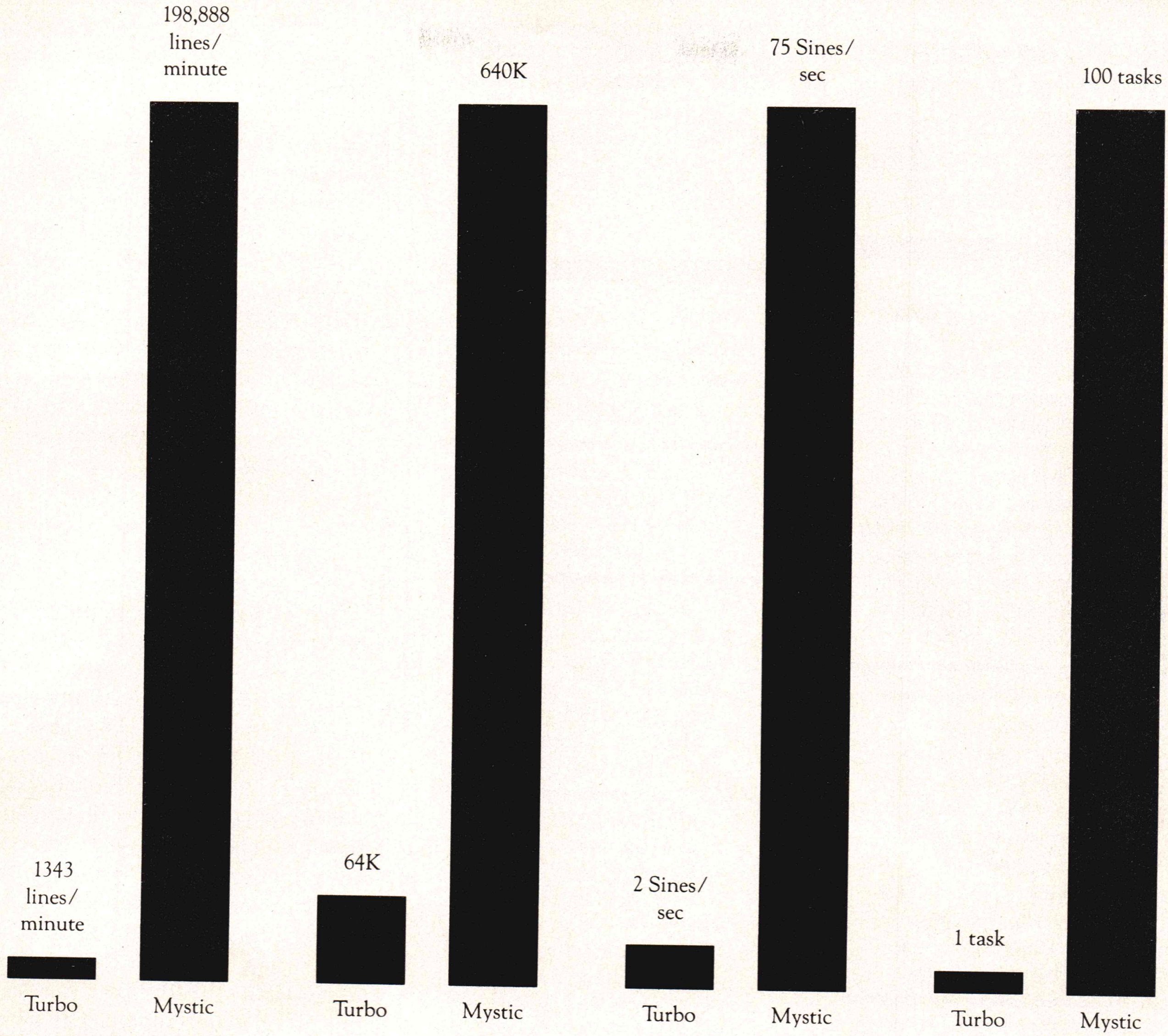
```
TwoToThe(k)
   if k = 1 then return 2
   let t = TwoToThe(k div 2)
   let t = t * t
   if odd(k) then
      double(t)
   return t
```

Although these are recursive functions, no level of recursion actually does anything until the next lower level has finished. Therefore, the variable t needn't be local to the function; each completed level may leave its result in a global t for the next level to work on.

Barker did our TwoToThe(k) function in Pascal, implementing t as a many-digit integer expressed as an array. If you give it 9865 digits, he says, you can evaluate 2 raised to the power 32767 in a couple of hours—and he sent a solid page of digits to prove it.

We liked his program so much we couldn't let it alone. We typed it in and then kept tweaking away at it, trying to make the multiplication of big integers (which dominates the run time) go faster. The much-abused result, so far from Barker's original that he might not recognize

MYSTIC PASCAL vs. TURBO PASCAL



COMPILER SPEED

Mystic Pascal uses a completely new compiler design to give extremely fast compilation of ISO Standard Pascal programs. It compiles each line of code the instant you key it into the Full Screen Editor. (Bar Graph is based on compiling a 179 line program with Turbo 3.0 on IBM PC.)

MAXIMUM CODE SIZE

Mystic Pascal lets you use up to 640K for your program's code and data. Your Personal Computer cost thousands of dollars, why settle for a programming language that only lets you use one tenth of its power?

ARITHMETIC SPEED

Mystic Pascal's floating point arithmetic does not use an 8087 but runs almost like it did! We use a special way of storing numbers in memory to achieve single precision arithmetic that is 5 to 50 times faster than other compilers. If you still need 8087 support, we'll be offering that soon.

MULTI-TASKING

The Mystic Pascal system is based on multi-tasking. The Editor, Compiler and other components operate concurrently. Up to 100 of your Pascal procedures may also execute concurrently. And they may communicate by passing messages through queues.

MYSTIC PASCAL FEATURES

- Interactive Pascal
- EXE file output
- 8086 optimized code
- Full Screen Editor
- ISO Standard Pascal
- Help Windows for Pascal Language
- DOS INTeRrupt calls

Mystic Pascal requires an IBM PC or compatible with 256K. Turbo Pascal is a registered trademark of Borland International, Inc.

QUANTITY DISCOUNTS

The price below includes the disk, 75 page manual and shipping within the US and Canada. We can optionally ship by UPS Blue Label (2nd Day Air) within the U.S. Foreign orders add 10% shipping, minimum $15.

Quantity	Price Each	2nd Day Air Price Each
1	$64	$4.00
2-4	56	3.00
5-10	48	2.50
11-20	42	2.00
21-50	38	1.50

USE YOUR CREDIT CARD TO ORDER NOW BY PHONE (505) 757-6344.

Mystic Canyon Software
P.O. Box 1010
Pecos, New Mexico 87552

Quantity ____________
Price ____________
Blue Label ____________
Total ____________

Name ____________
Address ____________
City ____________ State ______ Zip ____________

Sorry, CODs and Purchase Orders are NOT accepted. Payment must be in US funds on a US bank.

☐ Check/Money Order ☐ Visa ☐ Mastercard

Card ____________ Exp. ____________
Signature ____________

Circle **no. 79** on reader service card.

it, appears in Listing Four (page 22). It's for Turbo Pascal. Some of the tweaks may be beneficial only in the 8-bit version of Turbo, whose integer mul, div, and mod functions are decidedly lethargic.

Wired Tales

We asked if anyone had maintenance experiences to share, good or bad. It seems the bad experiences are easier to recall, or more fun to recount. Your responses put us in mind of *Weird Tales*, the old magazine of supernatural horror. Consequently, we decided to collect them together to form an anthology of tales of the *sub*-natural horrors of hardware. We call it *Wired Tales.*

No Parking

When he rested his head he got it in the neck!

After working with computers for eleven years and using my S-100 system for seven, *says Michael M. Dodd of Northbrook, Illinois,* I upgraded to a Compaq Deskpro. I'd had it just three weeks when I blew the power supply by turning the computer off, then on, too quickly. Are all switching power supplies this fragile? In seven years I never had a problem with the old Processor Technology supply.

Two weeks later I turned the Deskpro on and the 33Mb Rodime hard disk wouldn't run. It wasn't even turning. The dealer replaced it at no charge, then told me that I should *never* use the head-parking program. It seems this disk automatically parks its head when power is removed. Apparently, the supplied parking program runs the head into the bolts that mount the platters, jamming its rotation like a stick in the spokes of a bike.

I restored all my data from tape backup to the new disk and was back in business—almost. dBase III wouldn't run because of its copy-protection scheme. Yes, Ashton-Tate will replace the diskette, but this means I can never restore data from tape without uninstalling dBase III. I paid money for this?

The Victim

They rolled him for $60 and left him catatonic!

Let me tell you about my printer, *croaks Glenn English of Austin, Texas.* It's a Dataproducts B-300. I've run several trees of paper through it, but recently it has developed quirks.

The first was in the ribbon rollers. The ribbon lives in a plastic box. It's pulled out one end, dragged through the print head, and stuffed back in the box by a pair of rollers made of something that looks like transparent rubber. Mine wore out. Dataproducts carries them—for $60 a pair. They last three years, so that's OK, but the delivery schedule is not. Dataproducts insists it takes two months to process an order for parts they have in stock. (They have an 'expedited' service for *many* more bucks.)

I went to the local DEC office and asked for rollers for an LP-25 (a B-300 in a new box). They couldn't figure out what I was talking about. If I had the Dataproducts part number, just maybe they could cross-reference it, but they couldn't find it under their DEC model number. Luckily, a DEC maintenance tech happened by and found them for me. Price? $60.

But now the machine is down in a most peculiar way. It runs its power-on self tests OK and its internal ripple-print test as well. Put it on-line and its on-line LED lights up. But send it data and it just sits there.

Dataproducts does have a service department, but they sent me to TRW. TRW won't work on it because it is in my house. (I have no idea why.) One of their techs offered to swap boards with me if I'd leave a deposit of $700. When I plugged in the new board the printer went completely catatonic. That's where it stands now. In the future I think I'll stay away from Big Kids hardware. If my ProWriter breaks, I can get a new one for what it's going to cost to get the B-300 back on the air!

Satisfied At Last

The first one wanted too much and died!

I chose Shugart 851 drives, *Jay S. Rouman of Mt. Pleasant, Michigan, tells us,* because they were supposed to be the standard of the industry, and besides (I heard) they just never failed. Wrong. A plague of intermittent 'bad sector' errors settled in and I began looking for a cure. I've spent more than a few hours at the repair bench but thought that a disk drive should be best left to the folks at the factory.

They'd be glad to repair my drive, they said, for a flat fee of $250. It warmed my heart to know they'd suffer no financial loss; nevertheless I looked elsewhere. An independent repair outfit offered a flat fee of $200.

There was a message in this, I thought. With the help of a friend I tracked the problem to the index sensor. This circuit consisted of one IC and a few resistors. Resistors never fail, so I replaced the IC. Wrong.

Now, the resistors were in a pack, and replacement ought to have been a snap; after all, resistor packs are pretty simple. Wrong. This one had very strange values in a strange configuration and was custom made for Shugart.

I jumped for joy when the clerk in Shugart's parts department said the pack was just $1.50 and they'd be glad to ship it to me. Then he told me about the $50 minimum order. I complained that 32 and a third spare packs was a bit much for my two-drive system, but it got me only sympathy. He suggested I contact repair outfits or that I fabricate the part myself.

The homemade resistor pack (a wire sculpture of ⅛-watt resistors) got the drive in service, but I was steaming. I asked at several repair shops, but the best I could get was a promise to batch my order with one of theirs whenever they next might send one.

Then I happened on an ad for Hamilton/Avnet. That's a big company with many offices but still, the ad had the words *Shugart* and *parts* on the same page and gave a toll-free number. I called and it was wonderful! They had the part in stock. They would sell me a single one. They'd ship it UPS. They wouldn't insist on a purchase order or even a credit card—they'd bill me! And they were as good as their word.

Since then, Shugart has gone out of business (I like to think it's because of that $50 minimum-order policy) and I haven't needed Hamilton/Avnet again. Still, they're the first place I'll call if I ever need parts.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 190.**

Dr. Dobb's Clinic (Text begins on page 14)

Listing Two

```
100 REM Logarithm-based display of 2^K after D. Ross
110 DEFDBL A-Z : DEFINT I
120 REM if your LOG() is double, use LN10=LOG(10) here
130 LN10 = 2.302585093# 'natural log of 10
140 REM get max mileage out of short LOG()
150 DEF FNCLOG#(X#) = LOG(X#)/LN10
160 REM if your LOG() is double, use CLOG2=FNCLOG#(2#)
170 CLOG2 = .301029996# ' common log of 2
180 REM get integral power IK and set up
190 INPUT "Power of 2";IK : IF IK<=0 THEN END
200 DIM A$(IK+1) : FOR I = 0 TO IK : A$(I)="0" : NEXT
210 IZ = ASC("0")
220 A0 = IK * CLOG2
230 B = 1
300 WHILE B > 0
310     A = A0 + FNCLOG#(B)
320     IN = INT(A) : IF IN<0 THEN 399
330     ID = INT(10# ^ (A-IN))
340     A$(IK-IN+1) = CHR$(ID+IZ)
350     B = B - 10# ^ (IN - A0 + FNCLOG#(ID+0#))
390 WEND
399 REM early exit point
400 IF (ID AND 1) AND (ID<9) THEN A$(IK+1) = CHR$(ID+IZ+1)
410 FOR I=IK-INT(A0)+1 TO IK+1 : PRINT A$(I); : NEXT
420 PRINT : CLEAR : RUN
```

End Listing Two

(Continued on next page)

Dr. Dobb's Clinic

(Listing continued, text begins on page 14)

Listing Three

```
100 REM display 2^K by doubling
110 REM after Orin Safier
120 DEFINT A-Z
130 INPUT "Power of 2"; K
140 SIZE = 1+ INT( K * LOG(2)/LOG(10))
150 DIM DIGITS(SIZE)
160 DIGITS(1) = 1 :REM 2^0 = 1
170 MSD = 1 :REM count of live digits
200 FOR I = 1 TO K
210     C = 0
220     FOR J = 1 TO MSD
230             Q = DIGITS(J)
240             Q = Q+Q+C
250             C = - (Q>9) :REM "TRUE" is -1
260             DIGITS(J) = Q - (C*10)
270     NEXT J
280     IF C THEN MSD = MSD+1 : DIGITS(MSD) = 1
290 NEXT I
300 FOR I = MSD TO 1 STEP -1
310     PRINT CHR$(DIGITS(I) + ASC("0"));
320 NEXT I
330 PRINT : CLEAR : RUN
```

End Listing Three

Listing Four

```
program TwoPower;
{$A-  8-bit turbo, allow recursion }
{$R-  turbo, range checks off after test }

const maxdigit = 2048;

type
    digindex = 1..maxdigit;       { index/count over digits     }
    digit = 0..9;                 { stored digit of "number"    }
    addig = 0..99;                { temporary arith. result     }
{ type of a multi-digit decimal integer: it consists of an    }
{ array of digits with digits[1] being the LEAST signifi-      }
{ cant and digits[msd] being the MOST significant.             }
    number = record
        msd : digindex;
        digits : array[digindex] of digit
    end;

var
    n0, n1 : number;      { one is current result, depending  }
    which : boolean;      { ..on value of this switch }
    power : integer;
    timestab : array[digit,digit] of addig;

{ procedure to initialize timestab. see MULBYN.PAS for use}

procedure initab;
var x,y : digit;
begin
    for x := 0 to 9 do
        for y := 0 to 9 do
            timestab[x,y] := x * y
end;

{ procedures to manipulate "numbers" ...first, one to write }
{ a number to standard output as in write(n).               }

procedure Shownum(var n: number);
var j : digindex;
begin
    write(' ');
    for j := n.msd downto 1 do
        write(chr(ord('0')+n.digits[j]))
end;
```

(Continued on page 24)

Combine The Power Of C And Your IBM Mainframe

What are the advantages of running C on your 370?

C is a high level language offering speed, power and flexibility. When you combine this with the IBM 370, you've got an unbeatable environment for software development. In addition, C lends itself to many different types of applications, is highly portable to a number of diverse CPU architectures and is the basis for a wide variety of existing software. Clearly, the future of programming lies with C.

Why Choose Whitesmiths' C?

Experience. Whitesmiths has been developing C compilers for over seven years. We were one of the first to introduce C for the 370 and our experience is evident in the features we offer:

- Full implementation of the C programming language for the IBM 370
- Ability to run under VM/CMS and MVS, and to run interactively under TSO
- Ability to produce re-entrant code (discontiguous shared segments)
- Direct support for both EBCDIC and ASCII character sets
- Optional cross support for MS/PC-DOS, CP/M-86, CP/M-68K, VERSAdos-68K and CP/M-80
- Unlimited use of libraries in binary form
- Support for ROM-based programs
- UNIX-style compatible library
- Optional use of IBM or Whitesmiths assembler
- Support of full ISO Level 0 Pascal, extended to support separate compilation
- System and Machine Interface Libraries provided in source code form

Call Whitesmiths' toll-free number for more information
1-800-225-1030

INTERNATIONAL DISTRIBUTORS: UNITED KINGDOM - Real Time Systems, Ltd., Douglas, Isle of Man, 011-44-0624-26021; AUSTRALIA - Neology Pty. Ltd., Rosebury, Sydney 662-4111; JAPAN - Advanced Data Controls Corp, Toshima-Ku, Tokyo, 03-576-5351; SWEDEN - Unisoft AB, Goteborg, 011-46-31-125810; FRANCE - Cosmic SARL, Paris, 011-33-378-8357

Whitesmiths, Ltd. • 97 Lowell Road • Concord, MA 01742 • 617-369-8499 • Telex 750246

Circle no. 156 on reader service card.

Dr. Dobb's Clinic (Listing continued, text begins on page 14)

Listing Four

```
{ ...then one to set a "number" to a one-digit constant.     }

procedure Setnum(var n: number; d: digit);
begin
    with n do begin
        msd := 1;
        digits[1] := d
    end
end;

{ ..one to clear out k digits to zero (before multiply) and }
{ also set number to one-digit zero.                         }

procedure Zernum(var n: number; k: digindex);
var j: digindex;
begin
    with n do begin
        msd := 1;
        for j := 1 to k do digits[j] := 0
    end
end;

{ ...one to double a number by adding it to itself.     }

procedure Double(var n: number);
var j : digindex; dig, Cy : addig;
begin
    with n do begin
        Cy := 0;
        for j := 1 to msd do begin
            dig := digits[j];
            dig := dig + dig + Cy;
            if (dig < 10) then Cy := 0
            else begin
                dig := dig - 10;
                Cy := 1;
            end;
            digits[j] := dig
        end; {for j}
        if (Cy <> 0) then begin { number gets longer }
            if (msd = maxdigit) then
                begin
                    writeln('Oops, overflow on double');
                    Halt {turbo terminator}
                end;
            msd := msd+1;
            digits[msd] := 1
        end {if cy}
    end {with number n}
end;

{ ...and a general integer-multiply routine after that sent }
{ by David Barker: z gets x*y in the style of pencil and    }
{ paper multiplication.  Used in this program only to square}
{ a number as in z gets x*x.                                 }

procedure Mulnum(var x, y, z: number);
var
    i,j,k : digindex;
    xdig, dig, Cy : addig;
begin
```

```
    if ((x.msd+y.msd) > maxdigit) then
        begin
            writeln('Oops, multiply overflow!');
            Halt {turbo-specific termination}
        end;

    Zernum(z,x.msd+y.msd);
    k := 2; {in case x=0, which it can't in this program}
    for i := 1 to x.msd do begin
        Cy := 0;
        xdig := x.digits[i];
        if (xdig<>0) then begin
            k := i;
            if (xdig > 1) then
                {$I MULBYN.PAS mul-by-2-to-9 code}
            else
                {$I MULBY1.PAS mul-by-1 code}
        end {xdig > 0}
        else
            {x.digit[i] = 0, nothing to do}
    end {for i};
    with z do begin
        if digits[k]=0 then k := k-1;
        msd := k
    end

end;

{ In this program we have only 2 "numbers," n0 and n1.  We   }
{ start out using n0 and flop between it and n1 as we have   }
{ to square each temporary result.  The following procs use }
{ boolean "which" to apply Shownum, Mulnum, and Double to    }
{ "it" -- "it" being the one of n0, n1 currently holding the}
{ result of computations.                                    }

procedure Showit;
begin
    if which then Shownum(n1) else Shownum(n0)
end;

procedure Doublit;
begin
    if which then Double(n1) else Double(n0)
end;

procedure Squarit;
begin
    if which
        then Mulnum(n1,n1,n0)
        else Mulnum(n0,n0,n1);
    which := not(which)
end;

{Here is the recursive power-of-two algorithm, almost exactly }
{ as worked out in the text. }

procedure TwoToThe(K : integer);
begin
    if (K < 4) then begin
    { bottom of recursion, initialize number }
        which := false; { using n0 }
        case K of
        0 : Setnum(n0,1);    { doesn't happen in this prog. }
        1 : Setnum(n0,2);
        2 : Setnum(n0,4);
        3 : Setnum(n0,8);
        end
    end
    else begin
    { not at bottom, recurse to initialize, then square }
        TwoToThe(K div 2);
        Squarit;
        if odd(K) then Doublit
    end
end;
```

(Continued on next page)

Dr. Dobb's Clinic

(Listing continued, text begins on page 14)

Listing Four

```
begin
    initab;
    repeat
        write('Power of 2 (0 to end): '); readln(power);
        if (power > 0) then begin
            TwoToThe(power);
            Showit; writeln
        end
    until (power <= 0)
end.

{ --------- MULBY1.PAS ---------------------------------- }
{ code to multiply 1 * y.digits into z.digits, placed in }
{ an out-of-line include file for clarity }
begin
    for j := 1 to y.msd do begin
        dig := y.digits[j] + Cy + z.digits[k];
        if (dig < 10) then Cy := 0
        else begin
            Cy := 1;
            dig := dig-10
        end;
        z.digits[k] := dig;
        k := k+1
    end {for j};
    z.digits[k] := Cy
end {xdig=1}
\NP
{ --------- MULBYN.PAS ----------------------------------- }
{ code to multiply xdig * y.digits into z.digits when xdig>1}
{ placed out-of-line for clarity }
begin
    for j := 1 to y.msd do begin
{ note: in next statement, a direct "xdig mul y.digits[j]"  }
{ is replaced by a table lookup.  This reduces execution    }
{ time by 15% in 8-bit turbo but might be SLOWER in other   }
{ pascals or in PC turbo.                                   }
        dig := timestab[xdig,y.digits[j]] + Cy + z.digits[k];
        Cy := 0;

{ note: in 8-bit turbo the following ... }
        while (dig>9) do begin
            dig := dig-10;
            Cy := Cy+1
        end;
{ ...cut execution time by 50% compared to the more direct  }
{"if (dig>9) then begin Cy:=dig div 10, dig:=dig mod 10 end"}
{ ...this might NOT be true in other pascals or in PC turbo }

        z.digits[k] := dig;
        k := k+1
    end {for j};
    z.digits[k] := Cy
end {xdig>1}
```

End Listings

FREE **8087 VERSION with orders placed prior to Dec. 15.**

We've just invented an Algorithm Developer.

It just happens to be the best **programmable software calculator** on the market today.

```
amort.CFG
6-amrtp
months=months+1;pr=((pmt*12/int)/(1+int/12)
^(mths+1))*[1-(1+int/12)];@next
7-amrti
in=-pmt-((pmt*12/int)/(1+int/12)^(mths+1))
*[1-(1+int/12)];@next
8-sumr
prinpd=prinpd+pr;intpd=intpd+in;mths=mths-1;
@ifqt (mths==168);@goto(6)
```

A sample formula listing

PRO/SCI™ . **99 bucks** and a phone call.

Consider its performance.

- Programmable in simple BASIC—like statements
- Runs in *both* foreground *and* background modes
- Set up: Requires one key stroke
- Runs in background mode with most popular products (e.g. LOTUS, WordStar, dBase)
- Selects and moves rows of numbers *from* foreground document *to* calculator for *any* computation (e.g., standard deviation for 22 rows of numbers requires six key strokes)
- Returns results (at user's option) of all calculations to the foreground document
- Single-Step mode aids in algorithm de-bug
- Spreadsheet comparison
  — Greater range of formula capabilities
  — Access requires only key stroke
  — Transfers numbers to calculator without retyping
- Full formula line editing
- Ten formulas per formula file
- Unlimited number of formula files
- 8087 version available
- All calculations double precision
- Includes Accountant's style calculator with HEX, OCT, and BIN!
- Runs with SIDEKICK — in all modes

Functions
sin, cos, tan, asin, acos, atan, ln, exp, log, sqrt, abs, int, frac, round, mean, std, var, perm, comb, integrate

Operators
arithmetic: +, −, *, /
logical: #AND#, #OR#, #NOT#
relational: >, <, >=, <=, ==, <>
exponentiation: **, ∧
factorial: !

Programming Statements
Conditional and absolute looping and branching
Indexing by variable increment and test
Formula con-catenation (to 1000 columns)
Printing statements for line printer tabulations
Debug mode — single-step and display of all variables
Displayed decimal point control

Constants
pi, e, c, K, h, q, R, N_0, G, g

Send me ***PRO/SCI***™ for $99 and include 8087 Version at NO EXTRA CHARGE.
(Plus $5.00 shipping & handling)

☐ American Express ☐ VISA
☐ Money Order ☐ MasterCard
☐ Check

Card No. ____________
Expiration Date ____________
Signature ____________

Requires IBM PC or true compatible with 128K RAM and one disk drive and MS-DOS 2.1 or higher

NAME ____________
ADDRESS ____________
CITY/STATE/ZIP ____________
TELEPHONE ____________

California residents add 7% sales tax. Outside U.S.A. add $15.00. (If outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars.) Sorry, no C.O.D.

SYMSOFT
P.O. Box 4477
Mtn. View, CA 94043
Orders Taken:
Mon.-Fri.; 8am-4pm
800/227-6703
OUTSIDE CALIFORNIA
800/632-7979
INSIDE CALIFORNIA

Circle **no. 146** on reader service card.

Modula-2 versus Pascal for Microcomputers: An Update

by David W. Carroll

Even Niklaus Wirth admits that current implementations for Pascal restrict its usefulness to writing academic or "toy" programs. Will Modula-2 validate the structured Pascal model for serious software development?

David W. Carroll is the author of Telecommunications for the PCjr *(Micro Text/Prentice Hall) and* Programming with Turbo Pascal *(Micro Text/McGraw-Hill). He operates the High Sierra RBBS (209-296-3534), a bulletin board dedicated to the exchange of information about Turbo Pascal and other high-level languages.*

The May 1984 issue of *DDJ* (#91) featured the article "Introduction to Modula-2 for Pascal Programmers" by Hugh McLarty and David Smith[1]. In that article the authors covered the differences between Standard (ISO) Pascal and Modula-2. The present article is also aimed at programmers familiar with Pascal, but discusses more practical aspects of using Modula-2 in the microcomputer environment.

Developers of microcomputer software have been slow to adopt structured, high-level programming languages. PL/M and PL/I-80, Pascal and, most recently, C have been used to develop large applications programs, compilers and assemblers, but only a few machine-level systems have been written in these languages, notably, parts of CP/M in PL/M and all of Unix in C. There are two main reasons for this: first, some of these languages do not allow direct access to the machine; secondly, the code generated by available compilers is often inefficient.

To see how inefficient compiled code can be, you need only observe the difference in operating speed between Lotus 1-2-3, which is coded in assembly language, and Context MBA, which is coded in C. The speed problems experienced by Microsoft in using C to develop Windows finally caused them to revert to assembly language for large segments of the program.

PL/M and PL/I-80 are powerful languages, but have never caught on for development of commercial applications. PL/M has been confined to the Intel development system environment and PL/I-80 suffers from the same problems as its older sibling for mainframes, PL/I, namely, size and complexity.

Pascal was originally intended to be a teaching tool, a model language to demonstrate and foster structured programming in computer science courses. It was not meant to be a language in which large systems programs were written, and so does not include many of the features that are helpful, or necessary, for that task. Because the language is well structured and easy to learn, many software vendors extended it to include more features. However, because a formal standard for Pascal was a long time coming, many different versions were developed, resulting in the current incompatibilities between most extensions of the language.

C is a very popular high-level hanguage that does provide access to the machine level. It does not, however, enforce typing or offer the other structured concepts that Pascal does (for better or for worse). C code can also become very obscure (some may read elegant) and difficult to maintain.

Origins of Modula-2

Niklaus Wirth, who originally designed Pascal in 1971, created Modula-2 in 1977 and released the first technical report on the language in March, 1980. Modula-2 is based largely on Pascal and Mesa, a modular language developed by Xerox; the support for concurrent processing comes from an earlier Wirth language for real-time programming called Modula. Modula-2 eliminates most of the deficiencies of Standard Pascal. It also provides features that allow large systems to be broken down into small components that can be developed independently in a truly top-down fashion. Although other structured languages try to achieve this goal through the use of procedures and functions, Modula-2 achieves a much higher level of modularization and data isolation. Dr. Wirth noted at conferences at Stanford University and Sacramento, CA in June 1985: "I [developed] Modula-2 because all the compilers that were becoming available [for] Pascal were fine for toy [small] programs, fine for introductory courses . . . I [wanted] to show that structured programming languages [were] not just for the school. Their real value comes when you do big systems. For that you need efficient compilation."

Pascal's Limitations

If you program in Pascal, you are familiar with its limitations. It is true that extended versions of Pascal have been developed. There exist, however, no standards for these extensions, so that any application that uses them is not portable. A summary of the things missing from Standard Pascal is presented below:

1. Separate Compilation
2. Machine level interface
   - Bit-wise operators
   - Direct port and memory access
   - Absolute addressing
   - Interrupt structure
3. Dynamic strings
4. Multitasking
5. Procedure libraries
6. Definable abstract data types
7. Programmer definable scope of objects
8. An elegant way to exit loops before completion

The Benefits of Modula-2

The major advantage of Modula-2 is the concept of the module. Modules are stand-alone, self-contained units. Although they are compiled *individually*, they are not *independent*. A module consists of two parts, a definition module and an implementation module. The definition module declares all portions of the module that are visible from the outside, that is, all exported objects, as well as any imported objects that are needed to declare the exported objects. This allows the compiler to check in the library files for all required import symbols and to generate a symbol file for the new module. In addition, the inner components of the module are invisible to other modules, except for the items exported. The implementation module contains the actual program code. It is used to generate an object code file to be combined with other modules when the program is ready to be linked together.

The following example shows how Modula-2 can be useful in developing a large system. Suppose that a defense system is being built in which all files must be encrypted before being written to disk. With Modula-2, once the parameters have been specified, the encryption routine can be developed independently by a small group of programmers with a high security clearance. The rest of the module will be invisible. In the meantime, other programmers can develop the rest of the system using a dummy file access module in the library. When the encryption module is completed, it is included in the library, and the system program is re-compiled. A summary of the benefits of Modula-2 is presented below:

1. Modules
   - Separate Compilation with parameter checking
   - Control over visibility
   - Declarations vs. code sections
2. Definable abstract data types
3. Multitasking capability
4. Interrupt handling
5. Low-level definitions
6. Libraries
7. Larger programs

Using Modula-2

Modula-2 programs are generated in stages. First, the source file is compiled and the symbol (SYM) and link (LNK) files are produced. At this time, all external objects to be imported are checked. If they are not currently available in the library modules, they are flagged. When all modules have been compiled, the master program module is linked to all other needed modules. Finally, the output file is produced in whatever form is supported by the particular compiler (assembly, object [.COM or .EXE], or load file).

Pascal Programmers and Modula-2

Experienced Pascal (and C) programmers will have little difficulty acquainting themselves with the basic features of Modula-2 within a few hours; they should be familiar with the large number of standard library modules within a few weeks. However, the features that support the development of multitasking/multi-user programs go beyond the scope of Pascal and will naturally take longer to learn.

Differences from Pascal

An excellent reference book for programmers making the transition from Pascal to Modula-2 is *Modula-2 for Pascal Programmers*, by Richard Gleaves[2]. I will present here a brief summary of major differences between Pascal and Modula-2. However, a discussion of the added support for multitasking is beyond the scope of this article.

• Modula-2 specifically uses the ASCII character set. This

eliminates problems in porting to mainframes that use other schemes. In Modula-2 all identifiers are case sensitive. This means that GETDATA, GetData, and getdata are three different identifiers. Also, underscores are not allowed in identifiers. All reserved words must be written in upper case. Comments use only (* and *) as delimiters and may be nested.

• A new type has been added, the unsigned integer or CARDINAL type, used for counting numbers. Reals and Integers cannot be mixed in Modula-2, as the implicit type conversion has been eliminated.

• Program structure is very similar to that found in Pascal, but there are some minor differences. The reserved word BEGIN is no longer used to delimit blocks (except for procedure blocks); instead, all control structures allow multiple statements and require an END delimiter (except REPEAT . . . UNTIL blocks). The FOR structure has eliminated the DOWNTO keyword and now allows the step value to be declared. The declaration section of a procedure block no longer requires a specific order of declarations (in Pascal, LABEL, CONST, TYPE, VAR, PROCEDURE or FUNCTION). The last statement in a procedure must be an END with the same identifier as the procedure heading. FORWARD declarations are no longer required; procedures may be referenced before they are declared. PROGRAM declarations are no longer used; instead, the MODULE is the compilation unit.

• FUNCTIONs are not available in Modula-2; instead, a typed procedure is used that must include a parameter list. Procedures and typed procedures may have an empty parameter list. RETURN may be used to terminate a procedure, and a value may be included.

• A new LOOP structure is included, delimited with an END. The loop continues until an EXIT statement is encountered, at which point control transfers to the statement following the END.

• Formal open array parameters are allowed in procedures in the form ARRAY of T. The size of the array is determined at the time of activation by using the HIGH utility procedure.

• A significant change in Modula-2 is that all utility and file functions are now performed by library modules. Thus, even the simplest program module must IMPORT I/O procedure identifiers and EXPORT values.

Small Pascal programs can be converted easily to Modula-2. One major difference, as noted above, is the case sensitivity of Modula-2: all keywords must be written in upper case and other identifiers must be consistent. The program structure is nearly the same. Control structures simply do not use a BEGIN, although they require an END. IMPORT and EXPORT statements are required to access library routines like I/O. Functions must be changed to typed procedures.

Longer programs will benefit from separate compila-

tion of modules and the use of hidden objects allowed in Modula-2.

Modula-2 vs. Turbo Pascal

Ever since the release of Turbo Pascal two years ago by Borland International, there has been a resurgence of interest in the use of Pascal on microcomputers. With over 350,000 copies of Turbo Pascal sold, a large base of new Pascal users has been created. In fact, one prominent analyst of the software market has suggested that the influence of Turbo Pascal will cause Pascal or a Pascal-like language (Modula-2 or Ada) to become the language of choice for educators, engineers, software developers, and hackers, and possibly even to replace BASIC through the 1980s and 1990s. So, now that so many people are using Turbo Pascal, why should you consider switching to Modula-2?

Well, first let's get one thing straight: Turbo Pascal is not a true Pascal. As D.E. Cortesi pointed out in the July 1985 *DDJ*[3], Turbo does not conform to ISO Standard Pascal with respect to several required features. *[See this month's Letters section for response to Dave's column. Ed.]* In addition, Turbo has a large number of extensions designed to overcome many of the limitations of Standard Pascal. These two facts limit the portability of all programs written in Turbo Pascal.

Moreover, Turbo Pascal has limitations of its own. The most obvious one is that on the PC/MSDOS version both the program and the formal data area are limited to 64K. Of course, the heap area consists of all remaining memory, but not all data elements can be made up of pointer-variables!

Another limitation of Turbo Pascal is that it only produces .COM files, .CHN (chain) files, and overlay files. Furthermore, it is difficult to link separately compiled Turbo programs or separately assembled machine language programs. All parts of a Turbo program must be compiled together at the same time. There is no facility for separate compilation. If you are writing a 10,000 line program, this can be a significant factor.

What, then, is the value of Turbo Pascal. It is useful for writing small programs, but not large systems. It is also very important for demonstrating structured programming concepts, as Wirth originally intended. Borland is expected to release Turbo Modula-2 by the end of 1985 (Beta versions of the compiler for Z80 CP/M systems are currently being tested). The 8086 version will probably solve one of the problems with Turbo Pascal by allowing the use of the entire available memory in an IBM PC for both program and data storage.

The design of Modula-2 virtually requires separate steps for compilation and linking, as well as extensive library checking during compilation. This means that Modula-2 compilers will be somewhat slower to use for pro-

Tools for Modula-2 Programmers

Modula-2 compilers are currently available for many large and small systems. In addition to compilers, some companies (such as Information Systems, listed below) are developing lines of Modula-2 utilities. Modula Corporation is even selling a commercial version of the Lilith, the Modula-2 workstation developed by Wirth's team in Zurich. (Some 200 of the original Liliths were built.) The following list, prepared with help from contributors Brian Anderson and Ed Joyce, contains addresses of vendors of Modula-2 compilers and programming tools for microcomputers and some minicomputers. We expect to be able to add to this list in February, when we hope to review several Modula-2 compilers.

Modula Tools

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066
(800) 556-2283
Borland has a Modula-2 compiler for CP/M in Beta test, scheduled for release this year, with an MSDOS port to follow.

Fachbereich für Informatik
Universität Hamburg
Schluterstrasse 70
D-2000 Hamburg 13
West Germany
This is a source for a Modula-2 compiler for VAX/VMS systems.

Hochstrasser Computing AG
Leonhardeshalde 21
CH-8001 Zurich
Switzerland
01-47-55-48
Four graduates of ETH, where Wirth developed Modula-2, have written a Modula-2 compiler for Z80 CP/M systems.

Information Systems
1901 N. Fort Myer Drive
Arlington, VA 22209
(703) 522-8898
Thomas Woteki has developed a suite of Modula-2 programming tools using Logitech's Modula-2/86.

Interface Technologies
3336 Richmond, Suite 200
Houston, TX 77098
(800) 922-9049 and (713) 523-8422
M2SDS is a low-priced compiler for MSDOS systems; SDS-XP is a more expensive version. Preliminary reports indicate that these products still contain a number of bugs. ITS has established a bulletin board (713-523-7255, 300/1200 baud) to provide information and services for Modula-2 programmers.

Logitech
805 Veterans Blvd.
Redwood City, CA 94063
(415) 365-9852
Logitech has roots in early Modula-2 and Lilith development in Switzerland; we reviewed an early version of its Modula-2/86 compiler for MSDOS systems in February of this year. The latest version is 1.10. Logitech also publishes the *MODULA-2 Newsletter* to provide information about its product line.

Modula Corporation
1673 West 820 North
Provo, UT 84601
(801) 375-7400 and (800) LILITH2
Modula Corporation has compilers for Apple II, Lisa, Macintosh, and MSDOS. Its MacModula runs on 512K or 128K machines.

Maritime Infosystems, Ltd.
6660 Reservoir Road
Corvallis, OR 97333
(503) 929-2552
The Mosys Modula-2 System is a fully extensible, adaptable Modula-2 programming support environment for Sage, Stride and Pinnacle computers.

Dr. Josef A. Muheim
BBC AG
Abteilung ESL
Werk Turgi
CH-5401 Baden
Switzerland
Dr. Muheim has a Modula-2 compiler for PDP-1/RSX-11 systems.

P. Robinson
Computer Laboratory
University of Cambridge
Corn Exchange St.
Cambridge CB2 3QG
England
Mr. Robinson has a Modula-2 compiler for VAX/Unix systems.

Scenic Computer Systems, Corp.
14852 NE 31st Circle
Redmond, WA 98502
(206) 885-5500
Scenic distributes a Modula-2 compiler for the 68000.

BENCHMARK	Compile	Link	Execute
Turbo Pascal Ver 2.0	0:01	n/a	0:16
Logitech Modula-2 Ver 1.10	1:27	1:34	0:25*

All times in min:sec
*Includes loading runtime module
Test run on TAVA PC XT from hard disk.

Table
Compile, Link, and Execution Times for Sieve of Eratosthenes Benchmark

gram development than fast Pascal compilers like Turbo. Short programs for finding prime numbers (using the Sieve of Eratosthenes algorithm) are shown in Pascal in Listing One (page 34) and Modula-2 in Listing Two (page 34). The compile and link times for Logitech's Modula-2 compiler (Version 1.10) and Turbo Pascal (Version 2.0) are shown in the first two columns of the Table (above). Here, as expected, Turbo Pascal is the clear winner. The third column shows the time required for the execution of the compiled code. Although the code generated by the Logitech compiler seeems to run more slowly, part of the figure consists of the time required to load the runtime module, about 10 seconds. Thus, in execution time the numbers are actually comparable.

Notes

[1] H. McClarty and D.W. Smith, "An Introduction to Modula-2 for Pascal Programmers," *DDJ* #91, May 1984 pp. 22–27.

[2] Gleaves, Richard, *Modula-2 for Pascal Programmers*, Springer-Verlag, 1984.

[3] D.E. Cortesi, "Turbo Pascal vs. the Standard," *Dr. Dobb's Clinic*, *DDJ* #105 July 1985, pp. 12–18.

Further Reading

Cashman, M., "Extensions and Performance Improvements Keep Pascal Computing," *Digital Design*, August 1984, pp. 106–107.

Cooper, Doug, *Standard Pascal User Reference Manual*, Norton 1983.

Jensen, Kathleen and Wirth, Nikalus, *Pascal User Manual and Report*, Springer-Verlag 1974.

Joyce, Edward, *Modula-2: A Seafarer's Guide and Shipyard Manual*, Addison-Wesley, 1985.

Meng, B., "Ada and Modula-2: True Systems Languages?" *Digital Design*, August 1985 pp. 74–79.

Wirth, N. *Programming in Modula-2*, Springer-Verlag, 1983.

——— "Modula-2, An Overview," *Micro Cornucopia* #25, August-September 1985 pp. 76–79.

(Listings begin on next page)

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 191**.

Modula-2 vs. Pascal

(Text begins on page 28)

Listing One

```
{ Eratosthenes Sieve for Prime numbers in Pascal }
program prime;
const
  size = 8190;
var
  flags : array[0..size] of boolean;
  i, prime, k, count, iter : integer;
begin
  writeln('10 iterations');
  for iter := 1 to 10 do                    { do program 10 times }
  begin
    count := 0;                             { prime counter }
    for i := 0 to size do
      flags[i] := true;                     { set flags all true }
    for i := 0 to size do
      if flags[i] then                      { found a prime }
      begin
        prime := i + i + 3;                 { twice the index + 3 }
        {writeln(prime);}
        k := i + prime;                     { first multiple to kill }
        while k <= size do
        begin
          flags[k] := false;                { zero a non-prime }
          k := k + prime                    { next multiple }
        end;  {while}
        count := count +1                   { primes found}
      end;  {for, if}
  writeln(iter);
  end; {for }
writeln(count,' primes')                 { # of primes found in 10th pass }
end.
```

End Listing One

Modula-2 vs. Pascal

Listing Two

```
(* Compute a table of the first n primes numbers.  Print
  primes (optional). Use the the Sieve of Eratosthenes algorithm.*)

MODULE sieve;

FROM Terminal IMPORT WriteString, WriteLn;
FROM InOut    IMPORT WriteCard;

CONST size = 8190;

VAR flags: ARRAY[0..size] OF BOOLEAN;
    i,j,prime,k,count: CARDINAL;

BEGIN
  FOR j := 1 TO 10 DO
    WriteString('Pass = ');
    WriteCard(j,2);
    WriteLn;
    count := 0;
    FOR i := 0 TO size DO flags[i] := TRUE END;
    FOR i := 0 TO size DO
      IF flags[i] THEN
        prime := i+i+3;
        k := i+prime;
        WHILE k <= size DO
          flags[k] := FALSE;
          INC(k,prime)
        END;
        INC(count);
   (*   WriteCard(prime,8);
        WriteLn *)
      END
    END
  END;
  WriteLn; WriteCard(count,0);
  WriteString(' primes'); WriteLn
END sieve.
```

End Listings

We wrote the books on C . . .

Ecosoft's Eco-C88 C Compiler. $49.95

This has to be one of the best values on the market today. Eco-C88 has all operators and data types (except bit fields), has 8087 support, and comes with one of the most complete libraries available - over 180 functions including functions for color and memory files. The compiler produces fast, efficient code (as shown in the table) at one-tenth the cost of other compilers.

You also get a UNIX-like cc with "mini-make" that makes the compiler a snap to use. For a limited time, you will also get the Eco-C88 editor (described below) free of charge. Works with all IBM PC's and MSDOS machines.

	Eco-C88	L (1)	C86 (1)	MS (1)	MW (1)
sieve	12	11	13	11	12
fib	43	58	46	109	—
deref	14	13	—	10	11
matrix	22	29	27	28	29

1. Computer Language, Feb., 1985, p. 79. Reproduced with permission.

The employees at Ecosoft think everyone should explore what C has to offer. To further that goal, the books listed at right were written by the employees at Ecosoft to get you up and going as quickly as possible and to expand your understanding of C. Over 100,000 readers have placed two of the books on the best sellers list.

The ECOSOFT family of C products
ORDER FORM

☐ C Compiler $49.95________
☐ Programming Guide $20.00 ________
☐ Self-Study Guide $17.00 ________
☐ Programmer's Library $20.00 ________
☐ Program Editor $29.95 ________
Total* (Ind. res. add 5% tax)________
*Please add $4.00 for shipping.
Payment: VISA MC AE Check
Credit card expir. date________
Card # ________
Name ________
Address________
City, state ________
Zip ________ Phone________

Ecosoft, Inc.
6413 N. College Ave.
Indianapolis, IN 46220
(317) 255-6476 · 8:30-4:30

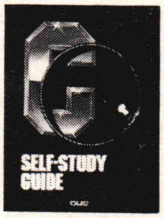
TRADEMARKS: ECO-C88, ECOSOFT; TURBO PASCAL, BORLAND INT'L.

The C Programming Guide (Purdum, Que Corp.). $20.00

This best seller walks you through the C language in an easy-to-read manner. All aspects of the language are covered with plenty of examples. Many of the error messages issued by the Eco-C88 compiler have page numbers that reference this book making the compiler and book perfect for the beginning C programmer.

The C Self-Study Guide (Purdum, Que Corp.). $17.00

This new book is designed for the individual that is learning C on their own. The book is filled with questions-answers and many examples about C and illustrates many of the tips, traps, and techniques in C that may take years to learn otherwise. Although written to complement the Guide, it can be used with any introductory C text.



The C Programmer's Library
(Purdum, Leslie, Stegemoller, Que Corp.). $20.00

This best seller is an intermediate text designed to teach you how to write functions in a generalized fashion. The book contains many useful library additions, including a complete ISAM file handler, plus sections on several advanced C topics.

CED Program Editor. $29.95

You're gonna love CED (pronounced "said"). It was written specifically for use with the Eco-C88 compiler. (CED creates a programming environment similar to that of Turbo Pascal.) You create the source file with CED, compile the program and, if there are any errors, CED automatically reloads the source file and places the cursor on the offending section of code. CED also has windows for editing multiple files, macro capability, and is configurable to suit your needs. CED is fast and has many features found only in editors costing up to ten times as much. Perfect for use with Eco-C88.

PC BRAND
We're NEW in this magazine but an Old Standby for IBM Program Developers

IF YOU PROGRAM IN 'C' DON'T TURN THE PAGE!

PADLOCK

RECOMMENDED!

Protect Any Program Using Anyone's Diskettes

You are a program developer seeking a low-cost method to protect against piracy? You are with a prominent company fearful of embarrassing lawsuits should employees copy programs illegally?

We have a devilishly clever solution for the two of you we call PadLock™. It gives you the keys to padlock any program you want—yours or someone else's—without buying a stockpile of expensive fingerprinted disks.

PadLock is software. One of its programs formats any manufacturer's disk, embedding your secret code in sectors the operating system will ignore. Another program installs this code in any program files you name, whether YourProg or WordStar™, or any .COM or .EXE files. From then on there's no picking the lock without the combination: the coded software together with a coded disk.

It's not despised copy prevention. Protected programs can be copied anywhere: to hard disks, to backup diskettes. It's a lock and key scheme. A protected program will just lock up at load time without a keyed diskette (such as its own) in one of the drives. The key disk may then be removed to free the drive.

Padlock is economical. You buy software that fingerprints 100 disks and their programs for $2.50 each, not costly fingerprinted disks you may never use at maybe $6.00 each.

Tell you what. PadLock keeps track of how many disks it has protected. Try it out. Format up to 5 test disks. If PadLock doesn't suit you, return it within 30 days for full refund.

Incidentally, don't look elsewhere. PadLock is available only from PC Brand.

Code	Our Price:
M0100 100 formats	**$250**

BRIEF

OUTSTANDING!

Is Anything But. A Whopper of an Editor

Dvorak in *Infoworld* said it for us: "the word is getting around...that Brief, The Programmer's Editor is simply the best text editor you can buy." With a name that belies its thoroughness, Brief™ has every feature you've ever contemplated for your editor-in-chief. Here goes:

Text, from keyboard or files, is housed in multiple buffers, and scrolled through one or more windows you open, close, resize. A text buffer may be called to different windows to view two areas at once. A change in one changes both. Text blocks may be marked for printing, writing to files, movement to scrap buffers for cut and paste into other buffers, or deletion, with as many "undo" levels as you want.

To find and fix, Brief has text search abilities rivaling "grep," with wild cards for matching, indifference to intervening characters, acceptance of character ranges, even multiple choice of patterns and their replacements.

If you use Lattice, C86™, or Wizard, and have 320k, you can compile your C program without ever leaving Brief. In fact, Brief will then find the lines with errors, and march you through the text for repairs.

On to macros! Parts of Brief were written with its own Lisp-like macro language. It has structure, 32-character variable names, conditional execution, loops, and you can actually read it! Nothing like the hieroglyphs we've seen elsewhere. A compiler comes with your copy, and prewritten macros in source to give you the flavor. Among the notables: one automatically indents C text; another turns Brief into a word processer with margin setting and wraparound.

There's a bulletin board with public domain macros contributed by a mushrooming cult of macromaniacs, and we've even hearing of megamoney prices for magnificance.

Then there's ... but, really, we're out of space. Who named this product, anyway?

Product Code: U0590 — Our Price: **$CALL**
List Price: **$195**
Needs 192k

C-SPRITE

Lattice's Own Symbolic Debugger for Lattice® C

This versatile companion to your compiler gives you the best of both worlds for an out of this world price. Hand it a COM or EXE file produced by the Lattice C™ compiler (using the -d option) and C-Sprite will speak your language: your function names, your variable names, your data types, and the line numbers from your source code. At the same time you can get a close-up view of machine addresses and machine-coded instructions, if you want to scrutinize just what machinations the compiler (or an assembler) contrived.

You already know how to converse with C-Sprite™ if you are familiar with Microsoft's Debug. Lattice began with that well-known command language, and added to it considerably: You can work with data in hex, as you might expect, but you can also differentiate between C's data types to cause the debugger to treat addresses as strings, long integers, even pointers, etc., both in display and entry.

C-Sprite can set breakpoints using symbols or addresses. You can submit clusters of commands to be executed at the breakpoints, or set commands that execute until a condition is met.

C-Sprite even has macros—use your source code variable names in a macro to dump the contents of entire C structures, for example. And you can debug through one of the COM ports with a second terminal so as not to disturb your program's display screen. What's more, if you link with Plink86™, C-Sprite can even tackle overlays.

Program doctors will find plenty of implements to rummage through in this kitbag.

Product Code: L2300 — Our Price: **$145**
List Price: **$175**

CURSES

A Screen Management Interface to Swear By

Curses manages the screen of the IBM PC in the same fashion as the curses utility of Unix and similar operating systems. Use it to adapt programs which call Unix's curses functions for screen management, and need the equivalent library when moved to the PC for re-compilation. Or use it when creating software on the PC to assure that it is Unix compatible.

Curses is a library of eighty-four functions and macros which can keep any number of screen images in memory. A screen may be full or partial size, and any can be summoned to the physical screen at your programs's command.

Within a screen, Curses employs a vast function set to get characters, wrap lines, scroll, blank lines, highlight— virtually any conceivable tool needed to update the screen. The product supports color, and all four memory models. Its input functions give you control over whether to echo each character to a memory screen. In keeping with the terminal orientation of Unix curses, the physical screen is repainted (at high speed) only when your program calls a refresh function.

Writing screen management code leads to unspeakable snarls and expressions. Swear off! Let Curses clean up your language.

Code:	List Price:	Our Price:
L0850	$125	**$100**
L0860 with Source	$250	**$200**

PC BRAND, Craftsman, PadLock and Toolbox TMs of PC BRAND / Unix TM Bell Laboratories / Lattice registered TM; and Lattice C, LMK, C-Sprite, CVUE and dBC TMs of Lattice Inc / dBASE TM Ashton-Tate / c-tree TM FiarCom / Pre-C, Plink86 TMs Phoenix Computer Products /MS TM of Microsoft / IBM registered TM Int'l. Business Machines / FirsTime TM Spruce Technology Corp. Inc. / WordStar registered TM Micropro / C86 TM Computer Innovations Inc / Access Manager, TM Digital Research / Brief TM UnderWare Inc.

FIRSTIME

Assistant Programmer for Hire. Real Cheap.

Not interested in interpreters? Then explore the other fast lane to coding: language-specific editors. Among them, FirsTime™ is the strongest C editor we have seen, with many more commands and built-in utilities.

FirsTime asks what program structure you want and sets it up for you. It could be "main", or "function", or "if-else" or "while" or any component of C. Ask for it and FirsTime fills in its skeleton, tending to semi-colons and curly braces.

It puts placeholders into these structures to prompt you to enter workable code—the initializers, conditionals and counters which govern an "if", for example. It checks that variables you enter have been declared, and highlights errors until fixed. Material can be pulled into ten buffers for movement elsewhere. Your last delete is held in one of them for "undoing". There's search and replace, writing of any marked block to a satellite file or the printer.

FirsTime can even transform one type of structure into another, such as "for" into "while" or "while" into "do".

Movement through the screen highlights the successive logical blocks of your program so that its structure is always apparent. And FirsTime can hide different levels of code detail to show you only the outer framework of your program.

FirsTime saves keystrokes to add speed, indeed, but more important, it lets you think at a higher level and makes a big contribution toward error-free code.

Product Code: E0460 — Our Price: **$229**
List Price: **$295**

LMK

A Unix-like "Make" Makes Light of System Building in Any Language

If you have ever built a complex system, you know the time loss and tedium of recompiling, rebuilding libraries and relinking modules because a snippet or two of code has changed. Batch files are no answer. You need batches of them to avoid redoing everything indiscriminately.

Instead, imagine making a change deep in a system, and simply telling Lattice's LMK™ to take over. No further thinking or keystrokes. LMK will rebuild your final product, however involved and complex, by doing just what is needed and no more.

How? You write a command file which expresses, bottom to top, all the elements comprising your system and all its dependencies: what gets compiled to make what object file using what options; what is built into libraries; what is linked into the final EXE file. Through the life of your system LMK keeps track of the last time every action was performed. Run LMK and, tracking each branch, it looks only for elements which changed later than a dependent element further along the branch, using date and time information in the file directory. Any source file newer than its object file, for example. Only those elements and their dependents are re-made. All other instructions are bypassed.

LMK does not care what programming language you use; it's not just for C. For that matter, LMK can apply to more than programming. It can be used for *any* set of tasks which can be accomplished through commands issued to the operating system.

Wherever your imagination leads you, LMK will find the shortest path to get the job done. Minimum time, minimum effort software.

Product Code: L2100 — Our Price: **$159**
List Price: **$195**

CODE SIFTER

Find the Fast Track for Your Program

You know it runs, but how fast? If it can't win, better not enter your program in today's horse race. Don't let it out of the stable until you've put it through its paces with our value- priced stopwatch.

Code Sifter finds the sluggish spots in your program. On its own, it will divide a COM or EXE file of any size into thirty-two equal partitions. Or specify the partition boundaries with addresses, or with symbols, if your linker has produced a symbol map. Then tell Sifter to run your program. It samples your object file at precisely timed intervals and counts how many times it finds the instruction pointer in each partition. Job done, it reports the number and percentage of hits in each.

You are in for some surprises when you discover how unbalanced the activity is likely to be, and that's why Code Sifter is so valuable. It profiles just where you can best spend time optimizing your code, or even converting to assembler subroutines.

Code Sifter has a number of monitoring options. You can tell it to include any combination of your program, DOS, and BIOS in its analysis. You can specify the sampling rate. You can tell Code Sifter the number of times to run a program, and between each run discard the less active ranges, and re-partition the hot spots, so that you zoom in ever-tighter on small areas of code. Right down to the last byte! Try it on the sample program that comes with your disk.

Code Sifter. It will give you the racer's edge.

Product Code: N3100 — Our Price: **$ 99**
List Price: **$119**

EXPORT EXPERTISE

PC BRAND ships anywhere. We'll prepare the export documents and ship to you or your agent by air parcel post, air freight or courier. Phone or Telex your order for speed. To pay by credit card, please provide number, date of expiry, name and billing address of card. Or wire funds to PC BRAND, c/o Chemical Bank, 126 East 86th St., New York, N.Y. 10028, Account No: 034-016058. We will ship immediately and acknowledge by Telex if a number provided.

CRAFTSMAN™ QUALITY SOFTWARE FOR 'C' & OTHER PROGRAMMERS

C-TREE

NEW FEATURES!

B-Tree File Manager, Source Code, No Royalties!

A b-tree can be infested with bugs, so before buying one, ask its age. In a stand of saplings, this one is a real cequoia. C-tree™ has been around since 1979. (It became Digital Research's Access Manager™). That means seasoned, sturdy code which hasn't cracked under prolonged and widespread use.

C-tree comes in C source code, revealing all you've ever wanted to know about how b-trees are written. Provided you bind it into your binary application, you can re-distribute c-tree without royalties.

And if all this is disappointing, now the good part. C-tree's design splits nodes to allow any number of users to access an index file simultaneously even when updates are in progress so that multi-user configurations and adaptation to networks are possible.

The latest version has new features: support of variable record length data files; multiple key indexes in a single physical file; MS-DOS and Unix record locking examples.

Thanks to source code which does not deviate from K&R, C-tree can travel. Binary has always meant finding a substitute file manager for yet another compiler, operating system, or computer; then changing all the function calls program-wide; then starting the whole testing process anew. That's over. Tests in many environments prove that C-tree gives your application a ticket to anywhere.

C-tree permits any number of keys for a data file, supports duplicate keys, alphanumeric or numeric, etc., etc.: it's a big product with everything you'd expect. Beyond that it is intelligently designed as both a high level set of ISAM routines to minimize your coding by handling all details of adding a record on its own, for example; and as low level operations which you can access directly. Either way C-tree maintains optimal index structures which will find a record amongst a million ten byte keys in no more than five disk seeks.

Product Code: F0660(	Our Price:
List Price: $395	$329

TEXT TOOLBOX

Tackles Text Tangles

Unix™ boasts a number of muscular utilities that are migrating to the PC world. Lattice has assembled a cluster of the most useful text management tools into a single package.

"Grep" looks for text patterns in any number of files. Want all occurrences of a global variable throughout a program system? Want to search all programs in a directory, down paths to other directories, or all files on a disk? Need to find all the function calls in an entire program system? Grep can do it with a powerful expression syntax that goes far beyond your text editor's search command.

"Ed" is similar to the well-known Unix editor. It offers search and replace with "grep's" syntax, block move, read and write, optional line numbering, append, insert, delete, and this unusual facility: you can instruct "ed" to apply a file of commands to any number of target files, even complicated changes and text additions, such as those created by "diff".

"Diff" compares text files line for line. Its output is a precise list of instructions telling what to do to make two files the same, a list which can be handed to "ed" to do.

Code:	List Price:	Our Price:
L2200	$120	$100
L2205 with Source	$240	$200

CVUE

Make Your Own Editor

CVUE is a low-priced screen-oriented text editor which does most of the things that a good editor should do, and boasts full DOS 2.0 directory path name support in reading and writing files.

It was written by the Lattice programmers who felt forgotten by the folks who write WP software. They needed easy entry of non-display characters such as control codes and escape sequences, not footnotes; indenting and undenting, not italics; pattern searching, not spell checking. So CVUE™ was born.

CVUE only supports in-memory text files, but with memory at today's prices, creating and maintaining files of over 500 K is practical. As compensation, CVUE is very compact and fast. It actually runs in computers with only 64 K of memory and uses no tediously slow overlays.

The power of CVUE is its ease of customization. And when you take advantage of the Source Code option, the resultant editor can be made truly your own.

Code:	List Price:	Our Price:
L2240	$ 75	$ 69
L2245 with Source	$200	$220

USED COMPILERS WELCOME

Trade In for the Latest Model Lattice C

Has your compiler run out of gas? Has your model been discontinued? Is it falling behind for lack of new parts?

Even if yours is in good shape, you have surely noticed there are more options and accessories produced to run with the Lattice C compiler than any other. Don't do without these additives any longer. It's time for new license plates. Trade in your original disks and manual of any of the compilers below and we'll send you Lattice's most up-to-date model.

From then on you will be adopted by Lattice for full, direct support by their technical specialists.

	Price:
Microsoft MS-DOS/PC-DOS C	$150
Computer Innovations C86, Mark Williams C, Digital Research C, Whitesmith's C	$200

PRE-C

Thorough "Lint"-like Analysis Now on the PC

Unix users long for a "lint" to give programs a thorough cleaning before they disappear into a compiler.

Pre-C™ looms larger than "lint." It finds problems your compiler won't. Problems that a debugger will have trouble figuring out. Even problems which will cause trouble with other compilers.

Pre-C finds all the syntactical tripwires that will blow out a compiler, sure, but it goes after subtler problems: code which will never be accessed, casts with suspect conversions, variables declared as external but never used, functions never called, obsolete usage (even C has changed), machine-dependent expressions with will inhibit portability.

Compilers work with one module at a time. They know nothing of other modules which only meet up at link time. Pre-C can look at *all segments of your program at once* and report to you any inconsistencies of inter-module references: conflicting data type declarations, parameter lists in function calls which disagree with the functions themselves in number or data type, declarations of external functions which differ from their definition.

Pre-C uses the Unix System III compiler standard to safeguard maxium portability anywhere in the C world. There are then plentiful command line options to advise Pre-C what to flag and what to forgive, useful during early coding when some functions are empty or incomplete. The output of each analysis can be filed for use with subsequent Pre-C runs, so work is not performed redundantly.

Pre-C lets you develop standing profiles of binary libraries. In any C program you subsequently write, Pre-C can use these profiles to make sure your calls to those libraries' functions are perfect.

This is a big product which will work miracles in speeding large system development. 128K minimum; 192K recommended.

Product Code: P0590	Our Price:
List Price: $395	$329

30-DAY MONEY-BACK GUARANTEE

We will return the purchase price[a] of any product in this space if you decide it fails to meet your needs except those coded "(" by requirement of their suppliers. We want you to be completely satisfied with your purchase. Go through the manuals, *try out the products*, and make a thoroughly informed decision whether they are right for you.

The only requirement for refund or credit is that we must *receive* your returned product within 30 days of its shipment, and it must in our judgment be in 100% resaleable condition.

[a] Not shipping and handling.

CALL FOR FREE CATALOG

There's more than this, but if we tried to include all the products PC BRAND offers, our ads would degenerate into the usual price list.

To see all, call for our catalog. Fancy it isn't. Simple keeps it changeable and printable for frequent distribution. But it will tell you about everything which could not be included in this space.

dBC

Switch from dBASE Language to C for Power, Speed

There are a lot of dBASE™ file users out there. Most of them just keep data bases and use dBASE's limited reporting facilities. They're not programmers, so they don't use the dBASE programming language. But they'd like more for their efforts, and that's a business opportunity.

dBC™ links C to dBASE. It creates and maintains files and their indexes which exactly replicate dBASE file design. So dBASE can read and update them. And the reverse. dBC can use any files created by dBASE. Now C and dBASE can operate on the same data bases interchangeably.

That opens up the widespread culture of dBASE installations to exploitation by C programmers. Now you can replace the resident dBASE language with the speed of C. And you no longer have to write every line of code, as in the dBASE language, because now you've unlocked C's vast storehouse of off-the-shelf libraries and utilities.

dBC's functions parallel all dBASE's file handling commands, many of them decomposed to give you closer-in control. The manual discusses each in detail, and demonstration source files on your disk show how every function is used. Use dBC for custom work for clients, or design generalized programs for manipulation and reporting of dBASE data bases.

Or use dBC on its own. It's a complete ISAM file manager for use with the Lattice C compiler whether or not dBASE will ever be used in tandem, has versions for all four memory models, and can have sixteen index and data files open at once.

Code & Version:	List Price:	Our Price:
L00II: Dbase II Compatible	$ 250	$225
LCCII: with Source Code	$ 500	$450
L0III: Dbase III Compatible	$ 250	$225
LCIII: with Source Code	$ 500	$450

TERMS AND CONDITIONS OF SALE

Licenses: Each price is for a license to use a product on a single computer and does not constitute its ownership. We will inquire for you about availability of multiple machine licenses at a single site. Except for those coded "#", products may be used to create programs for distribution without royalty payments or additional licenses, provided said programs do not substantially replicate the products themselves.

Compatibility: PC BRAND's standard products are designed to operate with the IBM® PC, XT or AT under PC-DOS and require no more than 128K of RAM unless indicated. Non-IBM machines using MS-DOS: If in doubt as to total compatibility, please determine from manufacturer.

Returns: Defective parts will be replaced. Products coded "(" are rendered unreturnable if sealed envelopes containing diskettes are opened. Otherwise, to return a product for refund or credit, call (212) 410-4000 for required prior authorization.

Payment: We honor MasterCard, Visa, American Express (no surcharge), wired funds, checks in advance (in NY State, add sales tax), and will ship COD (U.S. only) for cash, money order, or certified check (no fee). Purchase orders accepted from larger corporations and institutions at our discretion; terms net 30, 2% a month late penalty.

Shipping & Handling: U.S.: UPS Surface: 1st product $6, each add'l $3. UPS 2nd Day Air: 1st product $10, each add'l $4.50. UPS Next Day Air or Federal Express 1-2 Day Air: 1st product $18, each add'l $6. Federal Exp. Next Morning: 1st product $28, each add'l $7. International: Charges depend on destination and shipping method. $10 per shipping container for export or customs forms.

Circle **no. 108** on reader service card.

To Order Products or Catalog, Call Us at...

800-PC BRAND

That' (800) 722-7263. In NY State or Outside U.S. call (212) 410-4000.
PC Brand, P.O. Box 474, New York, N.Y. 10028
TELEX: 667962 (SOFT COMM NYK)

Prices, terms, and specifications subject to change without notice.

SAVE $5
Just tell us this code:
R11

Bit Manipulation in Modula-2

by Brian R. Anderson

Though Modula-2 provides a facility to manipulate data bits directly (via BITSETs), the methods available are not always convenient. Possibly, the most common use of this facility in high-level language programming is manipulating character data (bytes). A trivial example would be clearing the high bits of all characters in a WordStar file. In C, this is an easy matter of a single instruction (for each character):

```
ch &= 0x7F; /* bitwise and */
```

In Modula-2 the same operation takes five instructions:

```
I := ORD (ch);
   (* I is a CARDINAL *)
B := BITSET (I);
   (* B is a BITSET *)
EXCL (B, 7); (* Exclude bit 7 *)
I := CARDINAL (B);
ch := CHR (I);
```

This definition module is shown in Listing 1 (page 40) and provides the user (i.e., programmer) with an interface to the implementation module.

The first implementation module that I wrote was in Modula-2. My intention was eventually to write the code in assembly, but I wanted a working module to which I could compare the assembler version during the debugging phase. The Modula-2 implementation module is shown in Listing 2 (page 40). Despite the awkwardness of the Modula-2 syntax, this module was somewhat easier to write than the corresponding assembler module.

The Modula-2 compiler that I am using (Hochstrasser Z80) allows integration of standard MAC/REL files with Modula programs. This is accomplished by way of a conversion program. The assembly code is developed in the usual manner and then

The implementation module can be coded in assembly language for speed without affecting the definition module.

assembled using M80 or RMAC. A name translation file may be specified during conversion, at which time the REL file becomes an MRL file (Modula Object Code). (M80 shortens all identifiers to 6 characters and maps all characters to upper case, whereas this Modula-2 allows 24-character identifiers of both upper and lower case.) The assembler implementation module is shown in Listing 3 (page 44); the name conversion file is shown as Listing 4 (page 46).

With either implementation, the operation of clearing the high bit of

There has to be a better way! Though I knew from the outset that it was impossible to match C's simplicity, I set out to write a collection of procedures that would make it easier and more intuitive to perform common and important bit-manipulation operations in Modula-2

The first step was to write a definition module with eight common bit manipulations (set a bit, reset a bit, test a bit, shift left all bits, shift right all bits, AND/OR/XOR two bytes).

Brian R. Anderson, 2977 E. 56th Ave., Vancouver, B.C. Canada

characters in a WordStar file now becomes as simple for the Modula-2 programmer as it is for the C programmer:

```
Reset (ch, 7); (* reset bit 7 of ch *)
```

In fact, this is simpler and clearer than the C code to perform the same function.

Performance

From the point of view of the programmer the two implementation modules look identical. This is guaranteed because they both use the same definition module. Though no error checking is done, the procedures behave sensibly. If they are asked to set bit 9, for example, they simply circle around and set bit 9 – 8 (i.e., bit 1). You should note that these procedures consider the least significant bit to be bit zero.

The assembler module performs considerably better with regard to code size and speed. The Modula-2 implementation compiles to 555 bytes of code and 10 bytes of data, whereas the assembly implementation results in only 129 bytes of code and no data. The procedures in the assembler version are also 4 times faster (on average).

Conclusions

Bit manipulation in Modula-2 need not be as cumbersome as suggested by the definition of the language. The provision of assembly language interface for high-level language compilers allows significant performance improvements.

I would like to hear from other programmers if the facilities provided by the Bits module are useful enough to be considered for inclusion in standard libraries. Perhaps other procedures should be included (Rotate?). Let us know what you think before we approach the Modula Users Society (MODUS) with a proposal.

(Listings begin on next page)

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 192**.

Bit Manipulation

(Text begins on page 38)

Listing One

```
bits.def

DEFINITION MODULE Bits;
(* bit manipulation module *)
(* BA June 10, 1985        *)

   EXPORT QUALIFIED
      BYTE, Set, Reset, Test, ShiftLeft, ShiftRight, And, Or, Xor;

   TYPE
      BYTE = CHAR;

   PROCEDURE Set (VAR A : BYTE; bit : CARDINAL);

   PROCEDURE Reset (VAR A : BYTE; bit : CARDINAL);

   PROCEDURE Test (A : BYTE; bit : CARDINAL) : BOOLEAN;

   PROCEDURE ShiftLeft (VAR A : BYTE);

   PROCEDURE ShiftRight (VAR A : BYTE);

   PROCEDURE And (A, B : BYTE) : BYTE;

   PROCEDURE Or (A, B : BYTE) : BYTE;

   PROCEDURE Xor (A, B : BYTE) : BYTE;

END Bits.
```

End Listing One

Listing Two

```
bits.mod

IMPLEMENTATION MODULE Bits;
(* bit manipulation module *)
(* BA June 10, 1985        *)

   VAR
      cA, cB : CARDINAL;    (* cardinal value for parameters A & B *)
      bA, bB : BITSET;    (* bitset value for parameters A & B *)
      Result : BITSET;    (* result for and/or/xor *)


   PROCEDURE Set (VAR A : BYTE; bit : CARDINAL);

      BEGIN
         cA := ORD (A);    (* convert BYTE to CARDINAL *)
         bA := BITSET (cA);    (* 'coerce' CARDINAL to BITSET *)
         INCL (bA, bit MOD 8);    (* set the bit (make sure in range
                                                          0 --> 7) *)
         cA := CARDINAL (bA);    (* 'coerce BITSET back to CARDINAL *)
         A := CHR (cA);    (* convert CARDINAL back to BYTE *)
      END Set;


   PROCEDURE Reset (VAR A : BYTE; bit : CARDINAL);

      BEGIN
         cA := ORD (A);
         bA := BITSET (cA);
         EXCL (bA, bit MOD 8);
         cA := CARDINAL (bA);
         A := CHR (cA);
      END Reset;


   PROCEDURE Test (A : BYTE; bit : CARDINAL) : BOOLEAN;

      BEGIN
         cA := ORD (A);
         bA := BITSET (cA);

         IF (bit MOD 8) IN bA THEN
            RETURN TRUE;
         ELSE
            RETURN FALSE;
         END;
      END Test;
```

Lifeboat.™

C is the language. Lifeboat™ is the source.

Productivity Tools from the Leading Publisher of C Programs.™

The Lattice® C Compiler

The cornerstone of a program is its compiler; it can make the difference between a good program and a great one. The Lattice C compiler features:

- Full compatibility with Kernighan and Ritchie's standards
- Four memory model options for control and versatility
- Automatic sensing and use of the 8087 math chip
- Choose from the widest selection of add-on options
- Renowned for speed and code quality
- Superior quality documentation

"Lattice C produces remarkable code. . .the documentation sets such a high standard that others don't even come close. . .in the top category for its quick compilation and execution time and consistent reliability."

Byte Magazine

Lattice Library source code also available.

Language Utilities

Pfix 86/Pfix 86 Plus — dynamic and symbolic debuggers respectively, these provide multiple-window debugging with breakpointing capability.
Plink 86 — a two-pass overlay linkage editor that helps solve memory problems.
Text Management Utilities — includes GREP (searches files for patterns), DIFF (differential text file comparator), and more.
LMK (UNIX "make") — automates the construction of large multi-module products.
Curses — lets you write programs with full screen output transportable among all UNIX, XENIX and PC-DOS systems without changing your source code.
BASTOC — translates MBASIC or CBASIC source code directly to Lattice C source code.
C Cross Reference Generator — examines your C source modules and produces a listing of each symbol and where it is referenced.

Editors

Pmate — a customizable full screen text editor featuring its own powerful macro command language.
ES/P for C — C program entry with automatic syntax checking and formatting.
VEDIT — an easy-to-use word processor for use with V-PRINT.
V-PRINT — a print formatting companion for VEDIT.
CVUE — a full-screen editor that offers an easy way to use command structure.
EMACS — a full screen multi window text editor.
Fast/C — speeds up the cycle of edit-compile-debug-edit-recompile.

Graphics and Screen Design

HALO — one of the industry's standard graphics development packages. Over 150 graphics commands including line, arc, box, circle and ellipse primitives. The **10 Fontpack** is also available.
Panel — a screen formatter and data entry aid.
Lattice Window — a library of subroutines allowing design of windows.

Functions

C-Food Smorgasbord — a tasty selection of utility functions for Lattice C programmers; includes a binary coded decimal arithmetic package, level 0 I/O functions, a Terminal Independence Package, and more.
Float-87 — supports the 8087 math chip to boost the speed of floating-point calculations.
The Greenleaf Functions — a comprehensive library of over 200 routines.
The Greenleaf Comm Library — an easy-to-use asynchronous communications library.
C Power Packs — sets of functions useful for a wide variety of applications.
BASIC C — This library is a simple bridge from IBM BASIC to C.

Database Record Managers

Phact — a database record manager library of C language functions, used in the creation and manipulation of large and small databases.
Btrieve — a sophisticated file management system designed for developing applications under PC-DOS. Data can be instantly retrieved by key value.
FABS — a Fast Access Btree Structure function library designed for rapid, keyed access to data files using multipath structures.
Autosort — a fast sort/merge utility.
Lattice dB-C ISAM — a library of C functions that enables you to create and access dBase format database files.

Cross-Compilers

For programmers active in both micro and mini environments we provide advanced cross-compilers which product Intel 8086 object modules. All were developed to be as functional — and reliable — as the native compilers. They are available for the following systems:

VAX/VMS, VAX/UNIX, 68K/UNIX-S, 68K/UNIX-L

Also, we have available:
Z80 Cross-Compiler for MS- and PC-DOS — produces Z80 object modules in the Microsoft relocatable format.

New Products

Run/C — finally, a C interpreter for all levels of C Programmers.
C Sprite — a symbolic debugger with breakpoint capability.

Call LIFEBOAT: 1-800-847-7078. In NY, 1-212-860-0300.

YES! Please rush me the latest FREE Lifeboat™ catalog of C products.

*Name*__________ *Title*__________
*Company Name*__________ *Business Phone*__________
*Address*__________

Please check one of the following categories:
☐ *Dealer/Distributor* ☐ *End User* ☐ *Other*__________

Return Coupon to: Lifeboat™ Associates
1651 Third Avenue, New York, NY 10128

DD

Circle no. 125 on reader service card.

Bit Manipulation **(Listing continued, text begins on page 38)**

Listing Two

```
PROCEDURE ShiftLeft (VAR A : BYTE);

  BEGIN
     cA := ORD (A);
     cA := cA * 2; (* Shift Left is equivalant to Multiplication by 2 *)
     A := CHR (cA);
  END ShiftLeft;
PROCEDURE ShiftRight (VAR A : BYTE);

  BEGIN
     cA := ORD (A);
     cA := cA DIV 2;
     A := CHR (cA);
  END ShiftRight;


PROCEDURE And (A, B : BYTE) : BYTE;

  BEGIN
     cA := ORD (A);    (* both BYTEs must be forced to BITSET *)
     bA := BITSET (cA);
     cB := ORD (B);
     bB := BITSET (cB);

     Result := bA * bB;    (* AND the two bitsets *)

     cA := CARDINAL (Result);    (* force Result back to BYTE *)
     A := CHR (cA);
     RETURN A;
  END And;


PROCEDURE Or (A, B : BYTE) : BYTE;

  BEGIN
     cA := ORD (A);
     bA := BITSET (cA);
     cB := ORD (B);
     bB := BITSET (cB);

     Result := bA + bB;

     cA := CARDINAL (Result);
     A := CHR (cA);
     RETURN A;
  END Or;


PROCEDURE Xor (A, B : BYTE) : BYTE;

  BEGIN
     cA := ORD (A);
     bA := BITSET (cA);
     cB := ORD (B);
     bB := BITSET (cB);

     Result := bA / bB;

     cA := CARDINAL (Result);
     A := CHR (cA);

        RETURN A;
     END Xor;

END Bits.
```

End Listing Two

(Listing Three begins on page 44)

Bit Manipulation

(Listing continued, text begins on page 38)

Listing Three

```
;IMPLEMENTATION MODULE Bits;
;(* bit manipulation module *)
;(* BA June 13, 1985        *)
        NAME    ('BITS')
;
        .Z80
;
;   EXPORT QUALIFIED
;       BYTE, Set, Reset, Test, ShiftLeft, ShiftRight, And, Or, Xor;
;
        PUBLIC SET, RESET, TEST, SHL, SHR, ANDB, ORB, XORB
;
;   TYPE
;       BYTE = CHAR;
;
        CSEG
;
MOD8    EQU     00000111B       ;Mask to calculate bit MOD 8
;
;
;   PROCEDURE Set (VAR A : BYTE; bit : CARDINAL);
;
SET:    POP     IY              ;Return Address
        POP     BC              ;bit #
        EX      (SP),IY         ;BYTE's address <---> Return Address
        LD      A,MOD8          ;Make sure bit
        AND     C               ; in range 0 --> 7
        LD      B,A             ;'Safe' bit # in B
        XOR     A               ;Clear Accum. & CY
        CCF                     ;Set CY to make mask
        INC     B               ;Adjust count
FINDS:  RLA                     ;Rotate mask
        DJNZ    FINDS           ; until count is zero
        OR      (IY)            ;Set the bit
        LD      (IY),A          ;Return it to BYTE
        RET

;
;   PROCEDURE Reset (VAR A : BYTE; bit : CARDINAL);
;
RESET:  POP     IY              ;Return Address
        POP     BC              ;bit #
        EX      (SP),IY         ;BYTE's address <---> Return Address
        LD      A,MOD8          ;Make sure bit
        AND     C               ; in range 0 --> 7
        LD      B,A             ;'Safe' bit # in B
        XOR     A               ;Clear Accum. & CY
        CCF                     ;Set CY to make mask
        INC     B               ;Adjust count
FINDR:  RLA                     ;Rotate mask
        DJNZ    FINDR           ; until count is zero
        CPL                     ;Invert mask
        AND     (IY)            ;Reset (clear) the bit
        LD      (IY),A          ;Return it to BYTE
        RET

;
;   PROCEDURE Test (A : BYTE; bit : CARDINAL) : BOOLEAN;
;
TRUE    EQU     00000001B       ;
                                ;
TEST:   POP     HL              ;Return Address
        POP     BC              ;bit #
        EX      (SP),HL         ;BYTE <---> Return Address
        LD      IY,0            ;Clear a Pointer
        ADD     IY,SP           ;Make Copy of Stack Pointer
        LD      A,MOD8          ;Make sure bit
        AND     C               ; in range 0 --> 7
        LD      B,A             ;'Safe' bit # in B
        XOR     A               ;Clear Accum. & CY
        CCF                     ;Set CY to make mask
        INC     B               ;Adjust count
FINDT:  RLA                     ;Rotate mask
        DJNZ    FINDT           ; until count is zero
        AND     L               ;Check if bit set
        JP      Z,FALSE         ;If zero, return FALSE
        LD      A,TRUE          ; else, return TRUE
FALSE:  LD      (IY+2),A        ;Store function Return Value
        RET

;
;   PROCEDURE ShiftLeft (VAR A : BYTE);
```

(Continued on page 46)

Bit Manipulation (Listing continued, text begins on page 38)

Listing Three

```
;
SHL:    POP     HL              ;Return Address
        EX      (SP),HL         ;BYTE's address <---> Return Address
        SLA     (HL)            ;Shift Left (Arithmetic)
        RET

;
;   PROCEDURE ShiftRight (VAR A : BYTE);
;
SHR:    POP     HL              ;Return Address
        EX      (SP),HL         ;BYTE's address <---> Return Address
        SRL     (HL)            ;Shift Right (Logical)
        RET

;
;   PROCEDURE And (A, B : BYTE) : YTE;
;
ANDB:   POP     HL              ;Return Address
        POP     DE              ;'B'
        EX      (SP),HL         ;'A' <---> Return Address
        LD      IY,0            ;Clear a Pointer
        ADD     IY,SP           ;Make Copy of Stack Pointer
        LD      A,L             ;Move 'A' to Accum.
        AND     E               ;And with 'B'
        LD      (IY+2),A        ;Store function return value
        RET
;
;   PROCEDURE Or (A, B : BYTE) : BYTE;
;
ORB:    POP     HL              ;Return Address
        POP     DE              ;'B'
        EX      (SP),HL         ;'A' <---> Return Address
        LD      IY,0            ;Clear a Pointer
        ADD     IY,SP           ;Make Copy of Stack Pointer
        LD      A,L             ;Move 'A' to Accum.
        OR      E               ;Or with 'B'
        LD      (IY+2),A        ;Store function return value
        RET


;
;   PROCEDURE Xor (A, B : BYTE) : BYTE;
;
XORB:   POP     HL              ;Return Address
        POP     DE              ;'B'
        EX      (SP),HL         ;'A' <---> Return Address
        LD      IY,0            ;Clear a Pointer
        ADD     IY,SP           ;Make Copy of Stack Pointer
        LD      A,L             ;Move 'A' to Accum.
        XOR     E               ;Xor with 'B'
        LD      (IY+2),A        ;Store function return value
        RET

;
;END Bits.
;
        END                     ;
```

End Listing Three

Listing Four

```
(* Translation Table for converting Identifiers *)
(* from Assembler format to Modula format. *)
(* Table is used by module converter, which converts *)
(* standard REL file into MRL (Modula ReLocatable) Object file. *)

BITS        Bits
SET         Bits.Set
RESET       Bits.Reset
TEST        Bits.Test
SHL         Bits.ShiftLeft
SHR         Bits.ShiftRight
ANDB        Bits.And
ORBB        Bits.Or
XORB        Bits.Xor
```

End Listings

VEDIT PLUS

We have a Surprise for You.

NEW FEATURES MAKE IT BETTER THAN EVER

VEDIT is the most often reviewed and highly acclaimed text editor ever. And no wonder - we have listened to and supported our users for almost six years and have continually improved our products by incorporating hundreds of their suggestions. Now we have a revolutionary new version of VEDIT PLUS with all the program development features you've asked for.

The unique on-line interactive help is a first - it can be easily expanded or edited. You can create your own on-line help with menus and sub-menus for compilers, style guides, even non-computer subjects.

NEW FEATURES

- Memory management supports up to 1 M Byte
- MS-DOS, PCDOS pathname support
- Interactive on-line help
- Paragraph justification
- Create your own edit functions with macros
- Auto-execute macros as application programs
- Easy to use installation/customization program for keyboard assignment, screen attributes, etc.

The supplied macros are menu driven and immediately usable. Sort a mailing list. Perform a full screen file comparison and update one file from the other - merge the work done by two programmers on the same file. Replace the command prompt with a menu for new users. And an exceptional 8080 to 8086 translator is an inexpensive option.

Your own editing macros are easy to write and will automate otherwise tedious chores. You can use a macro the instant you type it or load it from disk - there is no time wasted compiling or linking. Macros are accessible from function keys - allowing you to create your own editing functions.

- Simultaneously edit up to 36 files of unlimited size
- Unlimited file handling - merge, split, extract, view, print and more
- Search with wild cards and patterns
- On-line integer calculator
- Complete macro programming language including:
  IF-THEN-ELSE and looping
  User prompts and input
  Algebraic expressions with variables
  Complete TECO capability
- Free macros

VEDIT PLUS isn't only for programming. The word processing features include word wrap, paragraph formatting with justification, printing and horizontal scrolling for spreadsheets. Converts WordStar files and edits dBASE™ source files. Add V-PRINT for fancier features such as automatic index generation and printer font support (incl. laser printers). Then add V-SPELL, the ultimate spelling corrector, and you'll have a word processing system that can do virtually everything you need.

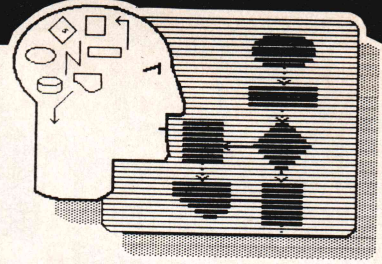
As a programming or writing professional, VEDIT PLUS gives you refinement, power and flexibility you will never outgrow.

VEDIT PLUS supports virtually every MS-DOS, PCDOS, CP/M-86 and CP/M computer and is a favorite among OEMs. VEDIT PLUS is not copy protected. Order direct or from any of our over 400 dealers. $225. (Discounts available for current VEDIT users.) Site licensing and educational discounts.

CompuView

1955 PAULINE BLVD., ANN ARBOR, MICHIGAN 48103
(313) 996-1299, TELEX - 701821

Circle **no. 122** on reader service card.



Zoomracks: Designing a new Software Metaphor

by Paul Heckel

The poet's eye, in a fine frenzy rolling,
Doth glance from heaven to earth, from earth to heaven;
And as imagination bodies forth
The forms of things unknown, the poet's pen
Turns them into shapes, and gives to airy nothing
A local habitation and a name.
A Midsummer Night's Dream

In my book *The Elements of Friendly Software Design*[1] I treat the design of applications software as a communications craft such as writing, filmmaking, or advertising. For over two years at Quickview Systems, we have been refining the ideas expressed in this book and putting them into practice in a product called Zoomracks. Zoomracks offers users a new environment and a new productivity tool. We have incorporated in this product four innovations in user-interface design:

1) A new computer metaphor: the rack. A rack is a familiar object. People know how items are organized on a rack. Consequently, a rack is an effective organizational model for a computer program. We think that Zoomracks will become a conceptual extension of the physical rack in the same way that VisiCalc became a conceptual extension of the physical spreadsheet.
2) A new viewing mechanism called Smart Zooms. Unlike windows, Smart Zooms shows the "big picture" rather than the details.
3) A method of using toggles and inverses to give even the novice a feeling of control in using the software.
4) A concept, called Fieldscrolls, that allows a user to construct a wide variety of database schemes by specifying a minimum amount of information.

Though the first two concepts were fundamental from the beginning, the last two evolved almost accidently into something more powerful than we had expected. This article is intended to explain what our thought processes were in designing and developing Zoomracks and not just what the product is.

The Opportunity

When you develop a new product, you should limit the number of fundamental goals that you hope to achieve. These goals will determine where thought, energy, and time are spent. They will determine what gets put in and what is left out—at least until the final stages of product development. In developing Zoomracks our fundamental goals were:

1. We wanted to present information on a computer screen effectively, compactly, and in a way that was independent of screen size[2]. Although Zoomracks is being introduced on the IBM PC, it was designed with the lap and hand-held computers of the future in mind. As semiconductor technology advances, computers will get smaller and smaller. By 1990, a circuit board the size of a credit card will hold 4 megabytes of RAM[3]. The microscopic size of electronic components will allow the manufacture of computers that are much smaller than those available today. Yet, the lap and hand-held computers will become practical and popular only if software is available that makes effective use of smaller screens. Programs that can operate only with larger and more expensive displays will naturally become less attractive. Of course, software that can utilize smaller displays must work at least as effectively with standard size computer screens.

The Japanese are experts at making things small. Their approach is to distill out only the essence and to eliminate what is not essential[4]. We tried to do the same thing in developing our software.
2. We wanted to provide an organization mechanism that was general enough to handle many kinds of information but was also simple and comprehensible to the user.

When there are pocket computers that have the power of a desktop computer, what will you want them to do? We think you will want them to keep track of the personal information that you want with you at all times: names and addresses, appointments, things to do, notes, memos, and the records of business expenses and automobile mileage that the IRS demands. Zoomracks was designed to organize and manage this kind of information. It was designed to be a general purpose tool as opposed to a specific application.
3. We wanted to give users a variety of ways to view their information.
4. We wanted Zoomracks to be easy to use.

. . . the idea itself probably is the most important element of the entire illustration. Certainly, if the idea is not good and if it does not interest and intrigue people, any other good qualities which the picture may possess will be lost because they will not be seen. It is an utter waste of effort to paint a beautiful, story-telling picture unless it is based on a good central idea—one which can be readily understood.
Norman Rockwell

It may be a good thing to copy reality; but to invent reality is much, much better.
Giuseppe Verdi.

You start with an idea or vision. From there you proceed to the techniques and technology required to bring that vision to life. A painter's first concern is not with his paints and canvas, but with the theme and mood of his picture. He imagines something in his own mind that he wants to convey to the minds of his audience. The tools of his profession are merely means (often uncooperative) to that end.

The programmer, like the painter, is a communicator. The same principles govern his creative efforts; his activities are arranged in the same hierarchy. The most important task for him is to choose a unifying idea. This unifying idea is like the spine of a book that holds all the pages together. It is only after this idea has been fixed that the programmer can think about techniques of presentation or technologies for communication. Our first concern, then, will be the unifying idea.

When an actor studies a play, he looks for a unifying concept around which he can build his entire portrayal. A program, too, must have a simple, communicable concept that holds everything together. The concept behind a spreadsheet program is the array of cells, each of which contains a number, an equation or text. Everything else is detail—important detail, but detail nonetheless. A good concept is not merely a reproduction of reality (like an image in a mirror), but creates a new reality. It might take time to attract an audience, but a good unifying concept will find resonance among users. VisiCalc, and more recently ThinkTank, opened up important new markets because they embodied such concepts.

The unifying concept becomes the basis for communication with the audience. Once chosen, it defines an easily intelligible framework into which everything else fits. It is the metaphor that strikes a familiar chord in the audience and allows them to talk about your product in familiar terms: "It's just like a spreadsheet." It generates word-of-mouth reports, creates new markets, and brings people in to purchase your product.

VisiCalc demonstrates how a good unifying concept works. At the point of sale, the question, "Can I do X with VisiCalc?" is transformed into "Could you do X on a manual spreadsheet?" In the office, "How would I do X with VisiCalc?" is turned into "How would you do X on a manual spreadsheet?" The user can do a lot just on the basis of information he already has.

Our unifying concept is a rack similar to magazine racks or the racks that hang next to factory time clocks. We started with racks because it was a simple, familiar concept that we felt could be developed both visually and in terms of complexity. First, we will describe a single rack, but imagine it as one of several side by side. Later we will describe how Smart Zooms display several racks on the screen at once.

Every single object shown in a picture should contribute directly to the central theme. All other things should be ruthlessly discarded.

Norman Rockwell

Consider a time-card rack. The first line of each card is always visible. Any card can be removed and the details examined. Cards can be inserted, removed, and moved into other slots in the same or different racks. Typically, cards in a rack have the same form (time cards), but are different in content (people and hours worked). They are probably arranged in some order (by employee number or name). Several time-card racks might be next to each other. These are the essential features of time-card racks. This is what people know and expect. This is the idea that serves as the unifying concept for Zoomracks.

Once we have selected the metaphor, we can begin to extend and transform our computerized racks to eliminate the limits that physical racks have. First, we can make racks arbitrarily long. A Zoomrack can hold 2 cards or 2,000, growing or shrinking to meet the number of cards in it. Second, a rack can automatically keep cards in order by the first field. People expect cards and lists to be ordered by the first field, as in a library catalog or a phone book. In a physical rack, inserting a new card in proper order requires moving all the cards below it to make room. A card inserted in a Zoomrack automatically goes to its alphabetic place. Third, although all the cards in a physical time-card rack are the same size, the cards in a Zoomrack can be of different lengths. At one extreme, a Zoomrack can hold a set of cards that contain only one brief item on the first line. If you remove the card, you see that the rest of it is empty. In this instance, the set of cards serves as a list of brief items of information. At the other extreme, a Zoomrack can be a file system. In this case, the first line contains the file label; when you remove the card, you find that the rest of it is filled by a lengthy document. Thus, you can use racks for lists, card files or for document files as you wish.

Exaggerate the essential and leave the obvious vague.

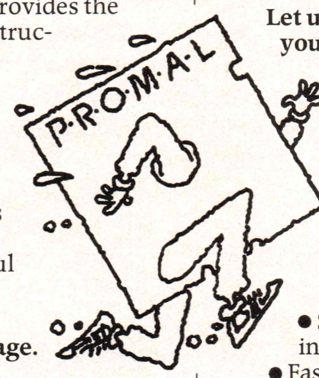
Vincent van Gogh

By striking one key, you can toggle between multicard mode, where you get a display of the first line of several

Now there is an even better structured, compiled programming environment than PROMAL. Introducing PROMAL 2.0 for the IBM PC, the Apple II, and the Commodore 64.

Until now, the best next language for the serious programmer was PROMAL.™ Now, it's the new PROMAL—PROMAL 2.0.

PROMAL 2.0 provides the same sophisticated structured programming environment, the same fast one-pass compiler, the same speed of execution, the same powerful commands of the earlier release —plus a host of useful new features.



Not just a language. A complete programming environment.

PROMAL—the PROgrammer's Micro Application Language—provides you with a complete programming environment, including a structured, high level language, a powerful program Editor, and a compiler that quickly turns your source code into compact, rapidly executing object code. Plus a library of integrated machine-language subroutines for frequently used tasks. And for the Apple II and the Commodore 64, PROMAL provides a DOS-like operating system Executive.

PROMAL 2.0— Even more of a good thing.

In addition to all of the features that have made PROMAL users declare it "the best language I've ever used," PROMAL 2.0 provides:

- Overlays that can be compiled separately for modular programming.
- Program size greater than 64k. (IBM PC only).
- True machine-to-machine portability.
- True 808X object code for the IBM PC.

Let us prove that PROMAL is your best next language!

Buy PROMAL 2.0 and try it for 15 days. If you don't believe it's your best next language, just return it for a full refund.

PROMAL Features

- Structured language with indentation.
- Fast, one-pass compiler.
- Simplified syntax.
- No line numbers.
- Multi-dimensional arrays, strings and pointers.
- Long variable names.
- Global, Local variables.
- Byte, Word, Integer & Real types.
- Decimal or Hex numbers.
- Functions and procedures with passed arguments.
- Built-in I/O library.
- Control Statements: IF-ELSE, IF, WHILE, FOR, CHOOSE, BREAK, REPEAT, INCLUDE, etc.
- Compiler I/O from/to disk or memory.

Executive*

Command oriented with line editing.
Allows multiple user programs in memory at once.
Function key redefinition.
Program abort or pause.
22 resident system commands.
Unlimited user-defined commands.
Prior command recall/edit.
I/O redirection to disk or printer.
Batch jobs.

Editor

Full-screen, cursor driven.
Function key controlled.
Line insert, delete, search.
String search and replace.
Block copy/move/delete/read/write.
Auto indent, undent support.
Edit after error.

Library

50 machine language commands.
Memory resident.
Call by name with arguments.
Formatted real output, string operations and much more.

*Apple II and Commodore 64 only. Requires one disk drive and 80-column card for Apple (IIe, IIc only).

1-800-762-7874
In NC: 919-878-3600

Systems Management Associates
3325 Executive Drive, Dept. D-1
Raleigh, North Carolina 27609

SMA

NEW for PROMAL The Graphics Toolbox*

Twenty fast subroutines for creating sophisticated, high-resolution graphics, including windows, clipping, scaling, and text-on graphics using scaled, rotated, user-defined fonts. $29.95.

*Available for the Apple II and the Commodore 64

Order Form

My system is (check one)
☐ IBM PC/100% compatibles ☐ Apple IIc/IIe
☐ Commodore 64/128

☐ Developer's Version—Compiler, Editor, Library, Demo disk, 280-page manual, (plus Executive for Apple and C-64) and stand-alone program generation. **$99.95 + 5.00 s/h.**

☐ End-User System for Apple II and Commodore 64—all features of Developer's version except stand-alone program generation. **$49.95 + 5.00 s/h.**

☐ Demo System—32-page "Meet PROMAL" manual and demonstration disk. **$10.00 + 2.50 s/h.**

☐ Graphics Tool Box for PROMAL—Available for Apple and C-64 only. **$29.95 + 2.50 s/h.**

☐ My check is enclosed.
☐ Please charge to my
__ Visa __ Mastercard

Card Number ________ Expiration Date ________
Signature ________
Name ________
Address ________
City, State, Zip ________

NC residents add 4-1/2% sales tax.
Foreign orders add $15.00 additional s/h.

Circle no. 104 on reader service card.

cards in the Zoomrack, and single-card mode, where you view the details of one card. This toggling becomes a reflex action, like switching a light on and off. By making it easy to flip cards in and out we hoped to make Zoomracks more like a physical rack.

Acting is a great profession as long as no one catches you at it.
Spencer Tracy to Burt Reynolds

One of our design objectives was to modify the perceptions of the user. Ordinarily, he perceives himself as giving instructions to a computer; we wanted him to perceive himself as manipulating real, familiar objects. If you see someone on television, that person seems real. From the evidence of your eyes you believe in his existence as much as you believe in the existence of the person sitting next to you in the same room. A good writer makes his characters seem real. You can probably conjure up a mental picture of Ebenezer Scrooge or Huckleberry Finn without much effort. They have been transported from the imagination of Charles Dickens and Mark Twain to your imagination. They may be fictional characters, but they are real to you.

Just as the job of a communicator is to create a reality in the mind of his audience, the job of the software designer is to create a reality in the mind of the user. When an author creates the "willing suspension of disbelief," the reader becomes oblivious to the fact he is reading a book. The software designer helps to suspend disbelief by minimizing the distractions caused by the mechanics of using the software.

One way to lessen distractions and make racks more real is to ensure that commands work the same in both multicard and single-card display modes (and, as we'll see later on, in multirack and single-rack modes). For example, the command for going to the next card in the rack—and thus the mental process required to perform this task—is always the same. When the form of a command depends on what mode you are in, you have to focus on the mechanics of the operation. This either actually disrupts your perception that the racks are real, or at least makes them psychologically more distant.

Each rack has a format that specifies its fields of information (e.g., name, phone, city). All the cards in one Zoomrack are of the same format, but different racks can have different formats. One Zoomrack could be used for names and addresses, another for appointments (sorted by date and time), a third for notes, a fourth for memos, and so on.

You can specify a format for any Zoomrack. More importantly, you can easily change it after the rack is already loaded with cards. This has two advantages.

1. People don't want to have to get it right the first time. You want to be able to try something, and when you see a way to do it better, change it. Adding, deleting, or moving fields should be easy.
2. People might want to concentrate information of special interest on the first line of a card where it can be viewed in multicard display mode. For example, if you have a card with the name and telephone number of an individual on the first line, you might want to move the name of the person's company or the date of an important meeting with him onto the same line.

One command toggles the display of labels on and off. This helps you use screen space efficiently because labels are left off most of the time. You only need labels when you are entering information or are first using Zoomracks. Usually labels are background noise: you are interested in your information, not the name of your information. Again, we are trying to provide the ability to view the essential.

Seeing is forgetting the name of the thing you see.
Paul Valery

Zoomracks consists of several racks. Each rack contains an unlimited number of cards. Each card has fields and a format that specifies the number of fields and how they are to be displayed. Each field has unlimited lines. Everything else is detail. (Our current implementation limits you to 10 Zoomracks, 29 fields per Quickcard, 80 characters per line, and the amount of RAM memory.)

Within this framework, Zoomracks offers three different capabilities: you can have short fields like those found in databases, text fields for multipage notes, and columns of information for forms such as sales orders or spreadsheets. The obvious way to do this is to have three types of field for columns, documents, and database entries and a bunch of rules for the user to follow. Unfortunately, this doesn't allow for easy use. While seeking a solution to this problem, we happened upon the Fieldscroll concept.

Each Quickcard is made up of Fieldscrolls that contain the text of the fields. To use a chemical analogy, if the Fieldscroll is an element, a Quickcard is a molecule, and a rack is a set of identical molecules. The format command lets you change the chemical structure.

The format command lets you specify how your Fieldscrolls are arranged, or, more precisely, which line each Fieldscroll is hung from, and, in the case of multiple Fieldscrolls on one line, the position of each within the line. Additional lines in a Fieldscroll are displayed successively on blank lines underneath. Thus, a Fieldscroll can be used in any of the three different ways mentioned above:

1. Database Mode: You display one field (typically, a name or phone number) on one line. Here the Fieldscroll consists of the single field on the single line.
2. Document or Text Mode: You display a document across the entire width of the display on consecutive lines. Here the Fieldscroll consists of the first line and the several lines for text underneath it.
3. Column Mode: You display several

narrow fields on the first line. Here each narrow field is a Fieldscroll. Now, if you hang blank lines from each field, these lines end up being the same width and you have transformed your set of narrow fields into columns.

You determine the meaning of your Fieldscrolls on a card. The format specification (where the name of the field and its location in the display are stored) is merely an aid to your understanding. Inconsistencies between cards and format (e.g., wrong field type, line too long) cannot occur.

Fieldscrolls were developed accidently. Zoomracks was evolving from an electronic Rolodex product into a multirack database product. We decided to add word processing. Because our database supported fields with subfields (or lines), we decided to treat the fields as text scrolls and the subfields as lines in the scroll. We had a move command in our electronic Rolodex that moved fields around to change the format. The quick and dirty (and, we thought, temporary) solution was to display additional lines of a Fieldscroll on succeeding blank lines that did not have Fieldscrolls on them. We thought that after we had used the product for a while we would get a better feeling for the problems involved and design a much better mechanism for specifying complex formats.

We ended up keeping the original solution (although we did polish it up a bit). The basic interface is simple. All the user really has to do to format a rack is to move Fieldscrolls around. If you put several on a line, they become narrow columns; if you put only one on a line, it becomes wide. You can insert blank lines to make them as long or as short as you want. The display is updated immediately.

Most importantly, virtually all requests that a user can make are legal and are transformed without loss of data into visible results that can be examined and modified. Any format can display any information because there are no limitations on type or size. As a result, several sources of error are eliminated. Specifically:

1. Cutting a card from one rack and pasting it into another is always a visible operation with no source of error even though the formats of the racks might be radically different.
2. Practically (but not strictly) speaking, there is no such thing as an improperly formatted external ASCII file.
3. The only important format operation is Move (move this Fieldscroll to there). It is always allowed and has no special cases. For example, a text field can be on a line with other fields. Other less important operations are: insert Fieldscroll, cut Fieldscroll, split line, join line, and edit Fieldscroll label.

When you use Fieldscrolls, a single phone number might appear in a sev-

eral line text field, or the text of a memo might be crammed into a phone number field, but these are not errors. No information is lost. The user can see the results and fix his problem just by changing his format. People find error messages annoying. They puncture the illusion of reality. We have attempted to make Zoomracks as flexible as possible to reduce the number of error messages with which the user must deal.

Of Mice and Menus

At the bottom of the hierarchy of things with which the designer of an effective user-interface must be concerned are what I call technologies for communication. These are the hardware and software technologies that help you communicate with a computer on a technical level: mice, voice recognition, menus, icons, high-resolution graphics, color, and so on. These technologies can help to improve the presentation of an idea, but can never transform a bad idea into a good one or a dull presentation into an interesting one. Good ideas that are well-presented are much more important than flashy technologies.

New technologies are merely tools that a software designer must manipulate in his attempt to communicate with the user a little better. They are never magic panaceas that eliminate the need to solve fundamental communication problems. Technologies can be seductive—they can distract the designer from his real concerns. Making a color film is easy. Making a *good* film—whether it be black and white or color—is hard. The same is true of software products.

Presentation and Staging

The task I am trying to achieve above all else, is to make you see.

D.W. Griffith

We have seen that the most important element in a good user-interface is the unifying idea and that the least important is the technology used for communication. In between these two in degree of importance is the technique used for presentation. In theater this is staging: how the scenery is arranged, where the players stand, and how they face each other and the audience. It is central to how an audience sees a play. Before the time of D.W. Griffith, people made movies by showing one continuous long-range shot of the action. To introduce a variety of perspectives into his presentations and to get us to see, Griffith invented the language of movies that we know today. He added close-up, medium-range, and panorama shots and edited them into a montage that controlled what his audience saw.

We need to find ways to add a variety of perspectives to our software products, just as Griffith did with movies. Currently, the main presentation mechanism is windows. Though many graphic systems give you the capability to zoom in and out, windows in text-oriented systems

generally have the same unchanging depth that movies had before Griffith. If early movies provided the unvarying remoteness of the long shot, windows generally provide the unvarying intimacy of the close-up.

I want to reach that state of condensation of sensations that constitutes a picture.

Henri Matisse

Zoomracks lets you have several racks on the screen at once. If you have a rack of appointments, another of names and addresses, and a third of notes, you might want to display them all at the same time. If windows are used to display files, you *do* see detail. However, when a window gets smaller or is hidden by other windows, you lose the overall view. Smart Zooms allow you either to obtain a panoramic view of several racks of information or to zoom in on any one of them to see its details.

Zoomracks lets you have up to ten racks on the screen at once, each of which is displayed with a Smart Zoom. If there are eight racks on the screen, then each of them is compressed to ⅛ of the screen width. Essential information is distilled from the cards on a rack and displayed. Vowels are deleted and words and fields are truncated to fit. You will find this works quite well when three or four racks are shown in 80 columns; it is useful even when all ten racks are on. You get a visual overview of the information in each rack and your imagination fills in missing details. The characters that are present suggest the full text—especially if you are already familiar with it.

The interface with the Zoomracks display consists of just three commands:

1. Typing any digit from 0 to 9 will turn on the corresponding rack (add it to the working set of racks displayed in multiple rack mode) and cause that rack to become the current rack (to be highlighted and be the object of any subsequent commands).
2. The Zoom command toggles between a panoramic display of all the racks that are turned on and a screen-sized display of the current rack.
3. One last command turns off the current rack (removes it from the working set of racks).

Just as all the commands work the same in single-card and multicard display mode, so all commands work identically in single-rack and multirack mode.

Zoomracks does not have a work line at the bottom of the screen. Rather, it positions the work line over the current field. This is likely to be controversial, because many people are accustomed to command lines. We feel, however, that most users focus their attention on the field where they are working and that something that pops out of a field seems more natural than something on a line at the bottom of the screen.

Toggles and Inverses

The user-interface for Zoomracks is table driven: for every key input there is a corresponding name in the command menu and a subroutine that is invoked. Reconfiguring this command network is a simple matter of restructuring the table. Consequently, during development we were able to make major changes in the user-interface without modifying the program proper. This enabled us to postpone the creation of a final user-interface until all the program functions were fully operational. During this time we used Zoomracks and carefully observed what we liked and did not like.

Quite late in the development we decided that the first target computer would be the IBM PC and that function keys should therefore be central to the user-interface. Early in development we had two toggle commands, Zoom Rack and Zoom Quickcard, that we liked very much. We assigned these two functions to F1 and F2. We assigned several other functions to the other function keys. At this time we wondered whether it would be possible to make all the function keys toggle. The answer was yes. As a result of making all the

function keys toggle we can tell the user, "You can hit any function key at any time and it will do something visible someplace on the screen. If you do not like what it does, press it again and it will undo what it did."

Part of the reason for doing this was that we realized that people often lose function key templates. We thought it would be nice to let the user hit any key until he found the right one.

In an early implementation of Zoomracks we had an exit command. When we looked at it closely, we found it was really three different commands: accept input (as in carriage return), return up the menu tree, and exit from a major command mode such as Edit or Format. It seemed that exit was a command concerned with the mechanics of doing things rather than with what was being done. We didn't like this. Therefore, we eliminated all the exits, largely by turning them into toggles. Both Format and Edit Format became modes that could be toggled on and off. Command menus such as Zoom or Modify became toggles that turned on a series of options.

As we were discovering the value of toggles, we realized that they were just a special type of inverse command. Almost everyone is familiar with the undo command, which cancels the effect of the last command (or in a few implementations the last several commands). We felt that every command could have an inverse or reversing command. For example, the command TAB=next Fieldscroll had its inverse BACKTAB=previous Fieldscroll. We placed inverse commands next to each other on the menu and used small symbols to identify them as pairs.

What is important is the feeling that users get when using toggles and inverses. Because things are reversible, they aren't worried about making a mistake. Just as important is the fact that toggles make them more adventurous; they know if something does not work, they can get out of it.

From our experience with Zoomracks in Alpha and Beta test we have learned the following:

1. Zoomracks is good at presenting information on the screen in a distilled form—particularly when you are dealing with several racks of information.
2. The commands for toggling between single-card and multicard modes and single-rack and multirack modes allow you to view your information in different ways.
3. Zoomracks provides a useful organizational metaphor[5]. It is a two-dimensional file system that can be extended to store macros, spreadsheets, communication cards, electronic mail files, and several other kinds of information. A Zoomrack can even be treated as a relational database table. Of these potential capabilities,

only macros are currently implemented in Zoomracks.

Zoomracks lets you define a rack of macros. The first character of the first field in each card is used for the macro name; another field contains its definition. Because a macro is a Quickcard in a rack, the user knows how to view, copy, delete, edit, and comment macros. He only has to learn the commands to execute and define them. Macros enhance the basic power of Zoomracks—the ability to work with several racks at once. For example, a macro can find and display all your appointments, to-do items, and notes.

4. Users of Zoomracks find that they end up doing many operations in multirack or multicard mode. Operations such as inserting a card, editing, or changing a format are done without the mental effort of going to single-rack or single-card mode.

5. As lap and hand-held computers become smaller and more powerful, we expect Smart Zooms to become particularly useful because they allow computers to make efficient use of smaller displays and the same user interface to be used on desktop computers as well.

Conclusion

The development of Zoomracks was not as linear and smooth as this article might suggest. Things that are clear and obvious now were fuzzy in the early stages of design. The thought processes described here were more important in recognizing, selecting, and polishing ideas, than in generating them in the first place.

I always have two things in my head—I always have a theme and the form. The form looks for the theme, theme looks for the form, and when they come together, you're able to write.

W. H. Auden

If you look at many of the early movies you will notice that the actors use exaggerated gestures. They seem strange and create a distance between us and what happens on the screen. We also see this when techniques appropriate for stage—where the actors must be at a distance from their audience—are used in film. Griffith realized that film was a more intimate medium and set about developing the techniques that are appropriate to it.

A similar thing happened in popular music when Bing Crosby recognized that the microphone opened up the possibility for more intimate communication with an audience. The techniques of the theater, where songs were belted out to large audiences, were inappropriate for radio and the phonograph. Crosby developed new techniques to go with the new medium. As computers get smaller and are used in a more personal way by more people, we have to develop techniques of making software that can reduce the psychological distance between computers and their users.

Zoomracks, Quickcards, Fieldscrolls, and Smart Zooms are trademarks of Quickview Systems. Rolodex is a trademark of Rolodex Corporation.

Notes

[1] Heckel, Paul, *The Elements of Friendly Software Design,* Warner Books, 1984.

[2] Tufte, *The Visual Display of Quantitative Information,* Graphics Press, 1983.

[3] Finke, Douglas, "Major Trends in Memory", *Transportable and Battery-power Personal Computers Proceedings*, *Future Computing*, 1984.

[4] O-Young Lee, *Smaller is Better: Japan's Mastery of the Miniature,* Kodansha Press.

[5] The idea of the *organizational metaphor* is well discussed by Chuck Clanton in "The Future of Metaphor in Man-Computer Systems" in the December, 1983 issue of *Byte*, pp. 263–280.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 193**.

You read Dr. Dobb's Journal And You Don't Subscribe?!

Save over $23.00 off newstand prices for 2 yrs.
Save over $10.00 for 1 yr.

Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, CP/M, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.

YES! Sign me up for ___ 2 yrs. $47 ___ 1 yr. $25

___ I enclose a check/money order
___ Charge my Visa, MasterCard American Express
___ Please bill me later

Name ______________________
Address ______________________
____

Credit Card ______________ Exp. date ______
Account No. ______________________
Signature ______________________

This offer good in U.S. only 3047

Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto CA 94303

WIZARD C

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

- All UNIX System V language features
- Support for 8087, 80186 and 80286
- Full library source code included
- Cross-file checks (full UNIX lint)
- Uses MS-LINK or PLINK 86
- ROMable data options
- In-line assembly language
- Cross compilers available
- Third party software available, including PANEL

The new standard for C Compilers on MSDOS!

Only $450.

(617) 641-2379

WIZARD
Systems Software, Inc.

11 Willow Court
Arlington, MA 02174

VISA MasterCard

Circle **no. 116** on reader service card.

EC Text Editor

. . . the Latest in Programming Environment Technology . . .
The DOS Interface — All Available Memory — Windows !!!

EC's DOS Interface is a major advance. It is far more sophisticated and useful than the "EXIT" to DOS provided by other editors.

Here's how you can use the DOS Interface:

1) Execute any DOS command you like.
2) Keep a history of all DOS output (you can even have EC automatically save it to a file if you want to keep a DOS Log.) EC displays the output in the DOS Interface. You can page through it, scroll it, etc.
3) You can re-execute or edit previous commands, even those very far back in the DOS history.
4) You can set a prompt, just like you would for DOS.

Here's how this can help you: You can run a database manager or a compiler inside the editor. Enter the command to execute your program just like you would in DOS. If you get error messages, exit the application, open a window on the error messages, and view the error messages as you edit your text. Jump instantly to the lines with errors. Correct all of the syntax errors after one compile; it's faster and more convenient than getting thrown back in your text time and time again as the compiler finds each error.

The DOS Interface captures ALL DOS output — even from your application. When your program terminates, and you want to edit your source code, you can still view your program's previous output! Especially useful as a debugging trace.

Read in large files — limited by your PC's memory. Move text between windows and buffers.

There's more: List command — like a random access Find, wildcards for Find/Replace/List, tab size dependent on file extension, auto-indent for C, on-screen file-comparison, Path support, support for read-only files, extensive screen and printer support, International and other extended ASCII characters, support for International keyboards, garbage pile of deleted text, columnar editing. Keystroke macros are created merely by hitting the keys — no codes to remember.

On-line tables: ASCII, extended ASCII, extended keyboard codes, operator precedence for C, DOS and BIOS functions, PC Memory Map. **On-line Help** and a **Tutorial.** EC was written with the DeSmet C compiler on a PC.

Please note: EC's low price belies its quality. EC stands with the best! The price is low because we want everyone to try EC and find out why it is superior to editors costing $200 and more!

EC is not copy-protected. **30 day money back guarantee.** Full-featured demo: $5.00.

EC Editor	$ 49.50
DeSmet C Compiler	99.50
DeSmet Debugger	50.00
BASIC__C Library	175.00
Lattice C Compiler	350.00
PC-Lint	89.50

EC and BASIC__C run on PC compatibles.
We use and support all products we sell.

C Source
12801 Frost Road
Kansas City, MO 64138
(816) 353-8808

Shipping: $5.00 for orders under $50.00. MC, VISA, COD. Site licenses available. OEM's, dealers welcome.

Other Products

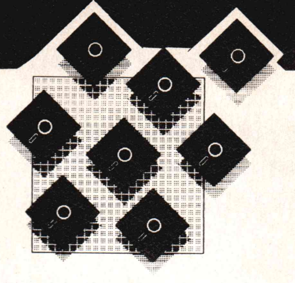
The DeSmet C Compiler — the fastest professional C compiler for the PC! DeSmet's program speed & efficiency are excellent. We chose DeSmet to develop EC and Basic__C, though, because of one factor: compile and link time. DESMET IS FOUR TIMES FASTER THAN LATTICE. 'Nuff said.

The Basic__C Library — the Convenience and Rich Function Set of BASIC for C.
The best way to move BASIC code over to C. The manual is regarded as the best tutorial for programmers moving from BASIC to C. You get a version for Lattice, DeSmet, and CI. No royalties. Includes source.

PC-Lint — A top quality lint checker at a quarter the price of the competition. Save hours of debugging by catching subtle bugs like function return mismatches.

Circle **no. 29** on reader service card.

Programming Editors, Programmable Editors



As a programmer, you spend many hours with your editor. If it is a good one and well-suited to your style of editing, it can greatly speed your production of code. If it is a bad one, or forces you to adopt an alien style, it can impede your work, spoil your outlook, and raise your blood pressure.

Beyond the basic functions, it is a matter of opinion what features an editor should offer. To choose an editor that is right for you, you need to know both what the different editors offer and what you yourself want your editor to do. In this review I describe some of the features, strengths and weaknesses of ten editors for the IBM PC and PC clones. It is up to you to select the editor with the style, performance, and price that meet your needs.

Features

What features might be of importance to the programmer? The tables that accompany this review list and evaluate many of the specifications and features of the various editors. They are by no means comprehensive, but they do cover some important aspects:

Size (see Table 1, page 62)

If you are working on a machine with limited memory, such as the PCjr, you need an editor that fits. Similarly, if you use floppy diskettes, you probably want your editor to be compact, so that it can be placed on the same diskette as your compiler and linker. If you work with both CP/M-80 and MSDOS, you may prefer an editor available for both eight-bit and sixteen-bit machines. If you work with large files, you may wish to choose an editor with virtual memory.

Mark U. Edwards, Department of History, University Hall, Purdue University, West Lafayette, Indiana 47907.

Documentation and Help (see Table 2, page 62)

You may not care whether an editor comes with a tutorial. On the other hand, a manual that does not have a good index or fails to group topics conveniently will cause you to lose a significant amount of time when searching for a particular piece of information.

Editing Commands (see Table 3, page 62)

With some editors you perform most fundamental tasks while in edit mode: you move your cursor around the screen, insert, delete, and so on, with sequences of control, alt, and function keys. Other editors add a command mode. This is used to execute more complicated commands and even macro language programs. Of the other editing functions noted in the table, the ability to reconfigure the keyboard is particularly important. You may not like the way the editing commands are assigned to keys. If the keyboard can be reconfigured, you can change the assignments to reflect, say, your preference for WordStar's keyboard assignments. If the keyboard cannot be reconfigured, either you must employ a keyboard enhancer, like SuperKey or ProKey, (assuming that the enhancer is compatible with your editor and does not steal space you need for files) or you must adapt to the configuration of the editor.

Search and Replace (see Table 4, page 66)

Here there are many choices. Do you need an editor that can search from the cursor backward toward the beginning as well as forward toward the end of the file? Is the ability to search for a string as you type it, one character at a time, an essential feature, or just a nice extra? Wild cards in a search allow you to locate general textual *patterns* as well as specific strings. Regular expressions, familiar to users of the Unix utility grep[1], allow the user to specify practically any pattern, including alternate patterns. You can, for example, search for all occurrences of "int" or "long int" in one search. Furthermore, if the editor supports regular expressions in search-and-replace operations, you have a powerful translating tool. This enables you, for example, to replace all Pascal if-then statements with the equivalent C construct. In Table 16 (page 74) you can compare the various wild cards or regular expressions offered by the different editors.

File and Window Management (see Table 5, page 66)

Most, but not all of these editors, allow you to edit files larger than RAM. Some do this automatically, through some virtual memory scheme. Others require (or, if you see this as an advantage, allow) you to page a large file manually. It can be extremely helpful to be able to edit one file while looking at, and excerpting pieces from, another. You must consider, however, how many files you need to be working on, and looking at, simultaneously.

Text Formatting Commands (see Table 6, page 66)

For the most part, a programmer's editor need not be a full-featured word processor. Even so, some text formatting capability can be useful on occasion, for example, for adding

comments to code. One formatting command that is especially valuable for producing well-structured code is the ability automatically or manually to vary the line indentation.

Printing (see Table 7, page 66)
Do you need the ability to print while editing? Most editors allow you to print all or part of the active file. Some can print one file while you edit another.

Undo (see Table 8, page 68)
Some editors allow you to undo deletions. Some also allow you to undo commands other than deletions. Obviously, this can be a useful feature. If you can undo everything you do, you are protected from most mistakes. Such a capability can, however, eat up memory and slow performance.

Keystroke Macros (see Table 9, page 68)
If you have ever used one of the keyboard enhancers, such as ProKey or SuperKey, you are familiar with keystroke macros: a series of keystrokes is assigned to a single key; then, when that key is pressed, the keyboard enhancer plays back the keystrokes as if you were typing them in yourself. Several of the editors have this or a similar capability built-in.

Macro Language (see Table 10, page 69)
Keystroke macros can take you only so far. To automate complicated editing tasks, such as complex translations or the automatic balancing of opening and closing parentheses in C or Pascal, you need a *macro language* that includes full conditional branching and the ability to set, manipulate, and evaluate variables. Actually, a macro language can be used to extend the capabilities of an editor in whatever direction you choose. In effect, with a good macro language you can create your own, customized editor. This does mean, however, that you have to master another programming language. The macro languages incorporated in the editors under review are based on one of two models: some resemble variants of Lisp and normally must be "compiled," or at least "tokenized," before use; others resemble Digital Equipment Corporation's TECO and are made up of short one or two letter commands that can be created and executed on the fly. In my opinion, the Lisp-like style is easier to learn and to use, at least for complex macros. Yet, programmers who are familiar with the terse commands of a TECO-like editor can make real gains in speed of creation, at least for simpler macros. To help you decide which kind of language is best-suited to your needs, I have provided sample programs using the different macro languages (see Listing, page 83).

Subprocesses (see Table 11, page 69)
MSDOS 2.0 and above supports an exec function, that is, it allows one program to execute another program, even command.com. An editor that takes advantage of this capability allows the user to run a compiler, linker or the MSDOS built-in commands, either from the editor itself or while the

	BRIEF 1.3	EC 2.1	EDIX 4.02	EMACS 0.8	Epsilon 2.03	MIX 1.2	Pmate 3.37	VEDIT+ 2.01	XTC 3.0	XyWrite II plus
A. size of program	73K	60K	81K	237K	79K	47K	29K[3]	26K	100K	72K
B. "full system" size	170K	271K	106K	488K	118K	86K	29K[3]	51K	122K	106K
C. minimum memory	192K	128K	192K	384K	192K	128K	64K[3]	64K	192K	
E. DOS	2.0	2.0	2.0	2.0	2.0	2.0[4]	2.0[4]	2.0[4]	2.0	2.0
F. file size limit	disk	memory	250K	memory	disk	disk[1]	disk	disk	disk[1]	disk
G. maximum line length	512	1K	255	avail. memory	media limited	255	250	1000	255[2]	255

Notes:

[1] Must manually page larger files. Cannot page backwards.

[2] Set at startup.

[3] This is the basic model. In Version 4.0 there are three other models: Menu model is 33K, Fortran model 32K, and C model 55K. The C model requires at least 128K.

[4] Also has a CP/M version.

Table 1
Size

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. pages	223	36[1]	164	258	65	98	196	302	91	518
B. table of contents	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C. index	fair[2]	fair[1]	good	Y	fair[2]	fair[2]	none	fair[3]	good	good
D. reference card	Y	N	N	N	N	N	N	N	N	N
E. on-line help	excellent	excellent	good	fair	fair	fair	fair	fair	fair	fair
F. tutorial	43 pp.	on-line	on-line	N	on-line	N	N	37 pp.	N	157 pp.

Notes:

[1] The current manual (8-1-85) is atrocious, but the on-line help and tutorial are so good that a manual is hardly necessary!

[2] Largely a list of commands with little or no cross references or topical entries.

[3] Good index for VEDIT manual, no index for VEDIT PLUS.

Table 2
Documentation and Help

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. modes	Edit	Edit	Edit	Edit	Edit	Cmmd. & Edit	Cmmd. & Edit	Cmmd. & Edit	Cmmd. & Edit	Cmmd. & Edit
B. cursor movement	full	full	full	full	full	full	full	full	full	full
C. deletion	full	full	full	full	full	full	full	full	partial[1]	full
D. block movement	full	full	partial[2]	full	full	full	full	full	partial[1]	full
E. reconfigure keyboard	Y	partial[3]	Y	Y	Y	Y	Y	Y	N	Y
F. extensible	[8]	[4]	[5]	[8]	[5]	[5]	[7]	[6]	[7]	[7]

Notes:

[1] Deletion and movement of blocks is done by whole lines.

[2] Block copies and block moves leave newline characters before and after the inserted block.

[3] Single control keys and twenty function keys can be assigned macros.

[4] Keyboard macros that can be assigned to a select number of keys.

[5] Keyboard macros that can be assigned to any key.

[6] Macro programs that can be executed from the command line.

[7] Macro programs that can be executed from the command line or assigned to a select number of keys.

[8] Keyboard and macro programs that can be assigned to any key or executed from the command line.

Table 3
Cursor Movement, Inserting, Deleting

editor is still resident in memory. Of course, there must be enough memory left in your system for the second program.

Error Handling (see Table 12, page 69)

At the very least, you should expect your editor not to fail if you leave your drive door open. Also, if you attempt to quit the editor without saving your file, the editor should warn you that you will lose all your modifications if you proceed. This is error handling at a fairly elementary level, but there is some variation among the editors on this score.

Benchmarks (see Table 13, page 70)

Speed is not everything. Still, if you have to wait too long for routine editing tasks to be completed, you may become frustrated and more likely to commit errors. To test speed of execution I chose several tasks that a programmer might well undertake in a session. If your compiler says that there is an error on line 436 of your file, you want to get to that line as quickly as possible. So, using a dual drive Zenith 161 running MSDOS 2.1, I timed how long it took the editors to load a 28K text file and jump to line 436. I also counted the number of keystokes that were needed to accomplish this task. Then, I timed how long the editor took to write the file to disk with a backup. I also created a five hundred line file that simulated an assembly language program with comments on every line. Comments were preceded by text, spaces, and tabs in various combinations. To test an editor's search and replace command, I replaced the five hundred semi-colons with "REM." Next, after restoring the semi-colons, I had the program change all the assembly language comments

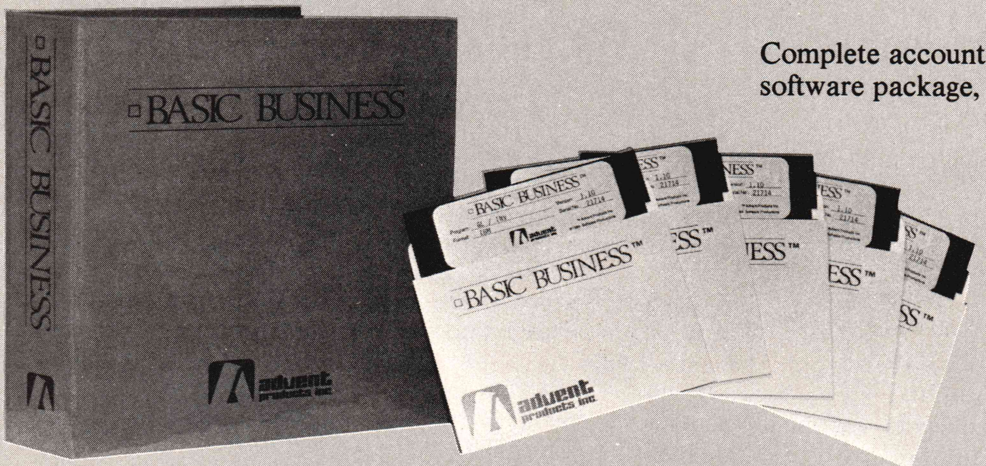
;<comment>

to C-style comments

/*<comment>*/

For some editors, this required writing a keystroke macro or a macro lan-

TIMELY REPORTS, ACCURATE RECORDS, RELIABLE INFORMATION......



Complete accounting software package, only **$89.95**

......*this is* ☐ BASIC BUSINESS

Price is nice - and so is a long list of features - but when you choose a software package to automate your accounting, don't lose sight of the basics of good business record keeping. Basic Business, an all-in-one accounting software package, offers an excellent price and extensive feature list without sacrificing these basic business principles:

Full Audit Trails - to give you complete confidence in the accuracy of your data and provide crucial backup information when you need it. And double entry accounting keeps your books in balance.

Complete Integration - provides efficient processing for all your business transactions, updating all of your accounting records instantaneously. You only enter the transaction once, and all supporting modules are updated automatically.

Flexibility - Basic Business can be adapted to your way of doing business, including balance forward or open item accounts receivable, departmental or consolidated general ledger, variable aging periods and easy entry of manually written checks and voids.

Extensive Reporting - each accounting module provides complete reports, including master lists, transactions, journals, statements and forms. In addition, all data files are compatible with Ashton-Tate's dBase II and dBase III, for the ultimate in custom reporting capability.

With some accounting software, even packages costing hundreds of dollars more, you must make detailed estimates and complex calculations for the maximum number of customers, transactions, inventory items, etc., before using the system. Then, when your business grows to exceed these original estimates, you must repeat this process.

Not with Basic Business! All data files are automatically initialized when you install the system on your computer. Files can grow dynamically as your business increases and are usually limited only by the amount of disk storage space available (Max. 65,000 records per file). No need for cumbersome reformatting once you exceed your original estimates.

FORMS ARE NO PROBLEM! Basic Business uses standard forms for invoices, statements, checks, purchase orders, etc., which may be ordered with your company name, adddress and logo imprinted.

Basic Business is one of a family of accounting software packages, and has sold previously for several hundred dollars per module. It has been improved, updated and re-packaged to sell at a market-busting $89.95 for all seven modules; General Ledger, Accounts Receivable, Accounts Payable, Payroll, Inventory Control, Sales Order Processing and Purchase Order Processing. A Point-of-Purchase module, which controls an electronic cash drawer and allows direct entry of transactions from your sales counter, is sold separately for $99.95.

SOURCE CODE AVAILABLE

Do you have a distinct accounting problem that off-the-shelf software won't handle? Special forms or statements? Don't write your own accounting system from the ground up - start with Basic Business. Call (714) 630-0446 for all the details on source code licensing.

Basic Business can go to work for you today and is available for all popular MS-DOS (IBM and compatibles) and CP/M-80 personal computers. Compare our price, features and attention to detail. There is only one choice . . . it's Basic Business.

Basic Business	**$89.95**
Point-of-Purchase module	**$99.95**
dBase II/III file formats	**$19.95**

Please add $2.50 for shipping.

Minimum Hardware Required for MS-DOS computers: Color or monochrome display, 256K memory, two DS/DD floppy disk drives (or 1 floppy disk drive and hard disk recommended for Sales and Purchase Order Processing), printer w/132 columns, MS-DOS (or PC-DOS) version 2.0 or later.

Minimum Hardware Required for CP/M-80 computers: 80 x 24 character display terminal, 64K memory, two 360K disk drives (hard disk recommended for Sales and Purchase Order Processing), printer w/132 columns, CP/M-80 version 2.0 or later.

Circle **no. 4** on reader service card.

SERVICES

File Transfer Service: Advent provides a service beyond the ability of any format conversion software! We can transfer files between MS-DOS/PC-DOS, CP/M and other operating systems in 300 different 3 1/2", 5 1/4" and 8" formats. Includes Apple and Mac, Apricot, Data General One, Kaypro 2000, Eagle, Epson QX-10 & PX-8 (ROM), HP-150, and North Star computers.

ENGINEERING SOFTWARE

ACNAP: A stand-alone Electronics Circuit Analysis Program for use with passive and active circuits consisting of resistors, capacitors, inductors, transistors, op-amps, FETs, etc. Features menu driven and very fast processing times with circuits saved to disk for later use or editing.
ACNAP (CP/M & MS-DOS) $69.95

DCNAP: Stand-alone DC circuit analysis program for use with passive and active circuits containing resistors, voltage sources, independent and dependent current sources. Fast, menu- driven program with circuit saved to disk for later use or editing.
DCNAP (CP/M & MS-DOS) $69.95

Plotpro: Scientific graph printing program. Prints on 80 or 132 column printer. Create linear, semi-logarithmic, and full logarithmic plots with one or two Y axes in auto or forced scale.
Plotpro (CP/M & MS-DOS) $69.95

SPP: This Signal Processing Program contains an integrated set of routines which analyze linear and non-linear systems and circuits and their effects on user specified time domain waveforms. Based on a 512 point Fast Fourier Transform and its inverse. Linear processing is in frequency domain and non-linear processing is in time domain.
SPP (CP/M & MS-DOS) $69.95

SOFTWARE UTILITIES

Autodiff: File difference detector. This program finds insertions, deletions, and changes between any two files. Autodiff can mark the file, display, or print the differences, and more!
Autodiff (CP/M) $29.95

CP/M DateStamper: Automatically stamp your files with the date it is created, last read, or modified. Works without a Real Time Clock , or with many clocks currently on the market. Utilities are included to allow copying, erasing, or renaming files based on time and date. A time logging utility is included to record computer usage for business/tax purposes.
DateStamper (CP/M) $49.95

Media Master +: Read and Write up to 75 CP/M, MS-DOS & TRSDOS disk formats on your IBM or look-alike computer. ZP/EM program is bundled with Media Master to allow CP/M programs to run directly on your MS-DOS computer. An $80.00 value.
Media Master + ... $59.95

Pack and Crypt: Two program set. Pack compresses and expands files on disk to save space. Crypt encodes files to provide security for sensitive data. Both are ideal for use with modem transfers.
Pack and Crypt (CP/M & MS-DOS) $24.95

Sidekick: One of the most popular programs ever written. Use Sidekick as a calculator, notepad, appointment calendar, auto dialer, ASCII conversion table and much more. On-line help if you forget any of Sidekick's many functions.
Sidekick (MS-DOS) $54.95

SmartKey II: New Release! Same great time saver as the original, and allows compiling of definitions you set up with your word-processor! Makes every software program you use easier. Can reduce keystrokes by more than 50% by redefining any key on your keyboard to be any combination of characters or commands that you desire.
SmartKey II $49.95

SmartPrint: A powerful add-on to SmartKey, SmartPrint is a vesatile writing tool designed to give you full access to your printer's features such as wide, bold, condensed, underlined, subscript, superscript, and more. Works great with programs like WordStar and others.
SmartPrint $29.95

Uniform: Your Computer can read and write up to 80 CP/M, MS-DOS / PC-DOS & TRSDOS disk formats. Versions available for most popular CP/M and MS-DOS computers. Specify your host computer when ordering.
Uniform (CP/M & MS-DOS) $69.95

XTREE: Directory maintenance program that graphically displays subdirectories and filename paths. Complete control of your directory including delete, rename, view, list or show. A must for your IBM or compatible.
XTREE (MS-DOS) $49.95

Super Zap: Disk patch and dump program. If you have used DU, you will love this menu driven marvel!
Super Zap (CP/M) $24.95

ZP/EM: Run almost any CP/M program on your IBM or clone. Use with Media Master or Uniform to allow programs on CP/M disk formats to run directly on your IBM or compatible computer.
ZP/EM (MS-DOS) $39.95

FX, QX-10, PX-8 - Epson Corp; CP/M - DRI; MS-DOS - MicroSoft; PC-DOS - IBM Corp.; dBASE II & dBase III - Ashton-Tate; WordStar - MicroPro; UNIX - Bell Laboratories; Apple - Apple Computer Inc.; Basic Business - Advent Products Inc.

PROGRAMMING LANGUAGES

C/80 Ver 3.1: Full featured C compiler and runtime library. One of the fastest on the market. Mathpak is included for true 32 bit floating point and signed integers.
C/80 Ver. 3.1 (CP/M) $79.90

C/NIX: Operating System Enhancement for CP/M. C/NIX gives your system many features in the UNIX OS such a hierarchical directory, I/O redirection, "pipes" & "filters" and command files. Uses only 2.3K of TPA and 42K of disk. Requires CP/M 2.x.
C/NIX $59.95

LISP/80: Experiment with the artificial intelligence language. Based on the INTERLISP dialect, LISP/80 offers over 75 built-in functions, including file I/O, and string operations. Complete with 36 page manual and demo programs.
LISP/80 .... $39.95

Toolworks C: This compiler is a complete subset of C. The two- pass compiler produces relocatable object files (.obj) which are compatible with the MS-DOS LINK program. Mathpak is included for true 32 bit floating point and signed integers.
Toolworks C Compiler (MS-DOS) $79.90

Turbo Pascal: Borland version 3.0. The best Pascal compiler on the market.
Turbo Pascal (CP/M & MS-DOS) $69.95

Turbo Toolbox: Set of 3 utilities for use with Turbo Pascal.
Turbo Toolbox (CP/M & MS-DOS) $54.95

Turbo Tutor: Teaches step-by-step how to use Turbo Pascal.
Turbo Tutor (CP/M & MS-DOS) $34.95

Turbo Graphics: Provides full graphics management for producing windows, pie and pie charts, circles and other geometric shapes with Turbo Pascal.
Turbo Graphics (MS-DOS) $54.95

TEXT EDITING

Punctuation & Style: Improves your writing by catching unbalanced quotes, parentheses and brackets, improper abbreviations, capitalization, sentence structure, much more. It's like having your own copy editor!
Punctuation & Style (CP/M & MS-DOS) $125.00

Word Finder: This powerful 90,000 word Thesaurus allows you to select the best word for the application. Works inside WordStar for greater ease of use. Instantly searches its dictionary, then displays synonyms, and automatically deletes the "wrong" word and replaces it with the "right" word. Requires 380K disk storage.
Word Finder (CP/M) $79.95

Wordpatch: Print files with tiny, compressed, wide, or wide compressed type faces, 5 sizes of italic, real superscripts and subscripts, and 6, 7, and 8 lines per inch spacing. No new print controls to learn. Supports most popular dot matrix printers. A must for WordStar users!
Wordpatch (CP/M & MS-DOS) $49.95

The Word Plus: The ultimate spelling checker. Not only finds misspelled words but shows you correct spelling options, shows the word in context, allows you to build dictionaries of special words you use, and much more.
The Word Plus (MS-DOS) $150.00

HARDWARE & SUPPLIES

Finger Print "Letter Writer": Unleash your Epson FX series printer. Add near-letter-quality print, IBM and/or Apple Graphics printer emulation, plus 16 other print functions! Three replacement chips quickly fit inside Epson FX series printers. Easy installation. Does not void printer warranty.
Finger Print "LetterWriter" $79.95

Diskettes, Double Density:

Maxell 10-pack w/ storage box:		3M box of 10:	
Single Sided	**$19.95**	**Single Sided**	**$22.95**
Double Sided	**$23.95**	**Double Sided**	**$26.95**

Economy Diskettes: package of 25 including tyvek sleeves.
Single Sided $29.50
Double Sided $31.25

Call or write for our FREE catalog

All items are warranteed for 90 days. 30 day money back guarantee if not completely satisfied. Guarantee for software applies only if diskette seal is intact. Visa and MasterCard are welcome. Please add 2.00 freight per total order and 2.00 for COD orders. California residents please add 6% sales tax. Prices, availability and specifications subject to change without notice.

CALL TODAY National **(800) 821-8778** California **(800) 521-7182**

Hours: Mon - Fri 8 am - 5 pm PDT

DEALER INQUIRIES WELCOME

advent products inc.
3154-F E. La Palma Av
Anaheim, CA 92806
(714) 630-0446

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. search backwards	Y	N	5	Y	Y	N	Y	N	N	Y
B. incremental search	Y	N	N	Y	Y	N	N	N	Y	N
C. query replace	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
D. wild cards	Y	partial[1]	Y	Y	N	N	Y	Y	Y	Y
E. regular expressions	Y	N	Y	Y	N	N	N	partial[2]	N	N
F. wild cards in replacements	Y	partial[1]	partial[6]	Y	partial[3]	N	N	N	partial[4]	N
G. undo replacements?	Y	N		Y	N	N	N	N	N	N

Notes:

[1] Has one wild card to match (and replace) a single character.
[2] Limited alternate pattern matching (A or B or C etc.).
[3] Can do a recursive edit upon finding an entry.
[4] Can place macros in the replace text.
[5] Searches only forwards but automatically wraps around to beginning of buffer and continues back to where it started.
[6] Only one wild card that can substitute for everything found in the search pattern.

Table 4
Searching and Replacing

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. number of buffers or files	Disk Capacity	5	12	avail. memory	Disk Capacity	2	11	37	11	2
B. number of windows	52	5	4	8	11	2	1	1	8	2
C. file merging	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
D. virtual memory	Y	N	N	N	Y	N[1]	Y	Partial[2]	N[1]	Y
E. Access to DOS file management	Y	Y	Y	Y	Y	Y	N[3]	Y	Y	Y
F. optional backup	Y	Y	Y	Y	N	Y	Y	Y	N	Y

Notes:

[1] Can page a large file in and out manually.
[2] Can page a large file in and out automatically.
[3] Available in version 4.0.

Table 5
File and Window Management

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. tab setting	variable	fixed	variable	fixed	fixed	variable	variable	variable	fixed	variable
B. margin setting	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
C. centering	Y	Y	N	Y	Y	Y	N[1]	N	Y	Y
D. word wrap	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
E. indenting	auto	auto	auto	Y	manual	manual	manual	manual	manual	

Notes:

[1] Manual gives a macro that could be placed on one of the select macro keys.

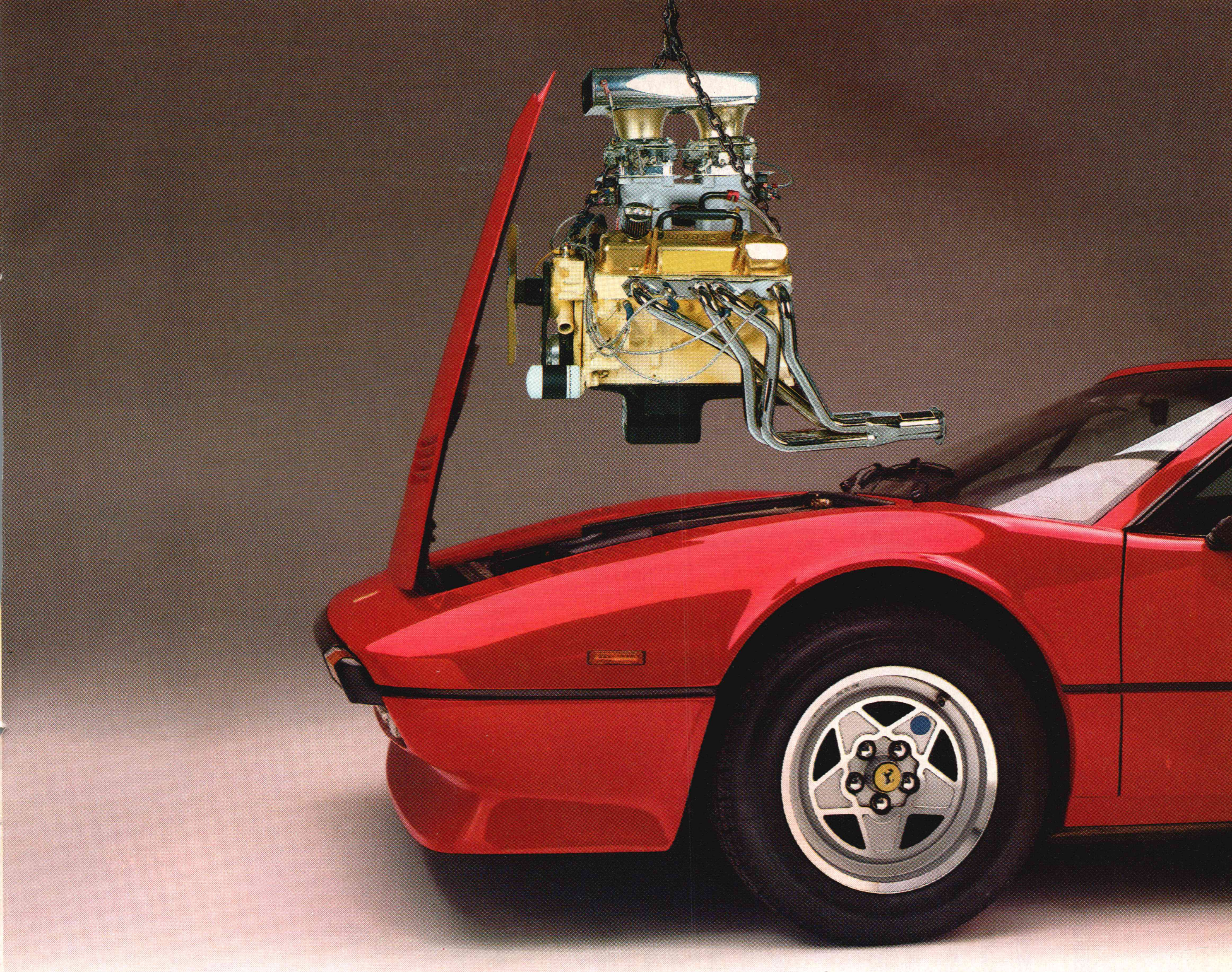
Table 6
Text Formating Commands

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. print active file	Y	Y	Y	Y[1]	Y[1]	Y	Y	Y	Y	Y
B. print while editing	N	Y	N	N	N	N	N	N	Y	Y

Notes:

[1] Must open a PRN file and write to it.

Table 7
Printing



AT™ Pfantasies for your PC or XT.™

Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."

If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

Programmers' Pfantasies™
by
Phoenix

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation.
For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

Circle **no. 91** on reader service card.

SEE US AT FALL COMDEX

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. undo deletions										
1. character	Y	N	N	Y	N	partial[2]	N	partial[2]	N	N
2. word	Y	partial[1]	N	Y	Y	partial[2]	Y	partial[2]	N	Y
3. line	Y	partial[1]	partial[3]	Y	Y	partial[2]	Y	partial[2]	Y	Y
4. block	Y	partial[1]	partial[3]	Y	Y	N	Y	Y	Y	Y
B. undo other commands	Y	N	N	Y	N	N	N	N	N	N

Notes:
[1] Only if using Alt-D command.
[2] Only if cursor has not been moved from the line.
[3] Only with certain commands.

Table 8
Undo

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. "on the fly"	Y[1]	Y[2]	N	Y	Y	Y	N	N	Y	N
B. save and restore	N	Y	Y	Y	Y	Y	N	N	Y	Y
C. pause for user	N	N	N	Y	Y	N	N	N	Y	Y
D. templates	Y	Y	Y	Y	Y	Y	N	N	Y	Y

Notes:
[1] Only one "on-the-fly" macro.
[2] For twenty function keys.

Table 9
Keystroke Macros (built-in keyboard enhancer)

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. model	Lisp	none	none	Lisp	none[1]	none	Teco	Teco	Teco	Teco[2]
B. full conditionals	Y			Y			Y	Y	Y	Y
C. "on the fly"	N[3]			N[3]			Y	Y	Y	Y
D. ease of use for simple macro	fair			fair			good	good	good	fair[4]
E. speed of creation for simple macro	fair			fair			good	good	good	fair[4]
F. ease of use for complex macro	excellent			excellent			fair	fair	poor[5]	poor[4]
G. speed of creation for complex macro	good			good			fair	fair	poor[4]	poor[4]

Notes:

1 The Beta version I saw had a C-like macro language with full conditionals. I did not test it extensively.
2 Not a complete macro language. Limited documentation and no telephone support.
3 Can be compiled and loaded automatically, making them almost "on the fly."
4 Some macros, especially complicated ones, are likely to be impossible. See note (1)
5 Cannot use complex boolean expressions in a single branch test.

Table 10
Macro Language

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. invoke DOS	Y	Y	N[4]	Y	Y	Y	Y[3]	N	Y	Y
B. concurrency	N	N	N	N	Y	N	N	N	N[2]	N
C. run compiler	Y[1]	Y	N[4]	Y[1]	Y	Y	Y[3]	N	Y	Y
E. Specify program memory	Y	N	N	Y	N	N	N	N	N	N

Notes:

1 Automatic compilation and flagging of errors, if any.
2 Can do fore- and background processes within the editor.
3 In Version 4.0.
4 The accompanying *Professional Writer's Package* can be used to invoke EDIX, WORDIX, INDIX, and other programs.

Table 11
Subprocesses

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. disk error recovery	good	poor[1]	good	poor[2]	good	poor[2]	none[3]	poor[2]	poor[2]	fair[4]
B. Abandon edit confirm	Y	Y	Y	Y	Y	N	Y	Y	N	Y

Notes:

1 See capsule review.
2 MSDOS does the error recovery. Ignore or Abort can lead to loss of edit.
3 Hung the machine completely. This was fixed in Version 4, but MSDOS still did the recovery.
4 Recovered OK, but said there was no such file on read.

Table 12
Error Handling

guage program; with others, a regular expression in a replace did the job. Next, after once more restoring the file to its original condition, I had the editor strip out all the comments along with any blanks or tabs preceding the comments. Finally, for those editors with a true macro language, I wrote a simple program to count opening and closing curly braces and tested the program on my IBM PC/AT. This last benchmark gives you a rough idea of both the speed (on an AT) and the intelligibility of the different macro languages (see Listing, page 83).

Miscellaneous Features (see Table 14, page 72)

I also tested whether the editor worked with Borland's SideKick and SuperKey. Here I assumed that the results could be generalized to other memory resident productivity tools. I have also noted whether IBM's extended ASCII character set can be accessed and displayed. Finally, since several editors include a built-in calculator, I have added that to my list.

Overall Evaluation (see Table 15, page 72)

The last table consists of my own subjective evaluation of these various specifications and functions. I explain some of my judgments in the capsule reviews of each editor (see below). One might expect that the price of an editor (both list and after discount, if it can be purchased at discount) should bear some relation to its features, but that is not always the case.

Editor-specific Comments

In each of the following capsule reviews (see Table 17, page 83, for specific product information) I focus on the particular strengths and weaknesses of a given editor along with selected special features. Together with the tables and the listing, these capsule reviews should give you a sense of what an editor is like.

One note about the versions reviewed. The tables display the specifications, features and performance of the *production* version of the editor that was available on August 1, 1985. Several of the editors had significant enhancements in the works, and I was able to see Beta test versions of the enhanced editors. In such cases I have added a paragraph on the enhancements to my capsule review. The tables, however, show the figures for the production version rather than the Beta version.

BRIEF (Solution Systems)

BRIEF's built-in commands are easy to use and to learn. Still, if you do not like a particular key assignment or the effect of a command, you can change it. The keyboard is fully reconfigurable and any macro program written in the powerful Lisp-like macro language can be assigned to

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. loading itself and test file and move to line 436 (seconds)	29	23	32	83	29	44[1]	16	28	86	35[10]
(keystrokes)	5	5	5	5	5	8	6	8	9	7
B. writing test file and backup (seconds)	30	[2]	18	24	9	[1]	23	30	23[3]	7
C. simple search and replace (seconds)	28	7	107[4]	16	23	49	38	17	108[4]	18[5]
D. change ASM comments to C (seconds)	47	244[7]	689[9]	36	53	341[8]	65	125	192[4]	328[4]
E. stripping comments (seconds)	30	—	355[4]	22	—	—	234	5	25 min.[4]	[6]
F. Scroll back 5 screens (seconds)	< 3	< 2	< 4	< 7	< 3	< 4	< 2	< 2	< 2	< 3
G. Brace count macro program (min:sec)	0:06	—	—	0:11	—	—	4:16	5:36	54:00	

Notes:

[1] MIX loaded only about 15K of test files.
[2] An earlier version of EC did this in 8 seconds; the current version was unable to do it at all. See capsule review.
[3] Does not do an automatic backup.
[4] Updates screen after every operation.
[5] Using "Change Invisible" command. With regular replace, which updates the screen, it took 125 seconds.
[6] I could not create a program that stripped comments from <space><tab><comment> lines.
[7] Done by holding the function key down until the keystroke buffer was filled. Could not search for either tab or space before the semicolon.
[8] Overflowed the keyboard buffer and put in extra, unwanted comment delimiters.
[9] This is the time for a "fully automated" keyboard macro. Using a more simple macro and repeatedly holding the envoking key down until the keyboard buffer was full, I did the conversion in 322 seconds. Updates the screen after every operation.
[10] Must specify page and line number on the page.

Table 13
Benchmarks

any key or invoked by name.

BRIEF has an outstanding undo facility. The default configuration allows the last 30 editing commands to be undone. This number can be raised to a maximum of 300 commands. Until you reach this maximum or run out of RAM, every command you issue can be undone. So if you make ten changes and then realize that the first one was an error, you can undo all the changes back to the mistake. This ability probably slowed BRIEF down on the benchmarks, but it meant that I was able to undo the results of each test with one keystroke! Needless to emphasize, this facility can save endless grief.

BRIEF has powerful search and translate commands. These use regular expressions and function much like the Unix grep utility (see Table 16 for a list of the expressions that are supported). For example, the string

```
m?l¦de
```

would match "male" or "made" or "mile" or "mode," and so on. The power of these operations is most apparent when you perform translation tasks. When you use regular expressions it is easy, for example, to change every Pascal if-then statement into its C equivalent. Unfortunately, BRIEF cannot search for strings that extend over several lines.

As with several of the other editors, you can, given sufficient memory, create from within BRIEF a subprocess that runs DOS. Within this subprocess any valid DOS command can be executed and other programs can be run. BRIEF comes with several macro packages that use this ability to compile source files. With one keystroke you can tell BRIEF to write the

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. compatibility with Sidekick and SuperKey	Y[1]	Y	Y	Y	N[2]	Y	Y	Y	Y	N[3]
B. IBM extended ASCII supported	Y	Y	Y	N	Y	N	partial[2]	Y	Y	Y
C. Built-in calculator	N	Y[4]	N	N	N	N	N	Y[5]	N	Y[6]

Notes:
[1] Must use -p flag on startup.
[2] Control-Alt, used to invoke Sidekick, is a prefix for some Epsilon commands. Does not work with SuperKey.
[3] Works with Sidekick, but not with SuperKey.
[4] Has logical, arithmetic, and shift functions as well as ASCII conversion.
[5] Has logical and arithmetic functions.
[6] Has four arithmetic functions.

Table 14
Other Features

	BRIEF	EC	EDIX	EMACS	Epsilon	MIX	Pmate	VEDIT+	XTC	XyWrite
A. price	$195.00	$49.50	$195.00	$495.00	$225.00	$29.95	$225.00	$225.00	$99.00	$295.00
B. discounted?	Y	N	Y	N	N	N	Y	Y	N	Y
C. ease of use	good	good	good	fair	good	fair	fair	good	good	good
D. documentation	good	good[1]	good	fair	fair	fair	poor	good	good	fair[2]
E. editing power	excellent	good	good	excellent	excellent	fair	excellent	excellent	good	excellent
F. formatting power	good	good	fair[5]	good	good	good	good	good	good	excellent
G. translating power	excellent	fair	good	excellent	fair	fair	good	good	good	good
H. file handling	good	good	good	fair	good	fair	good	good	fair	good
I. error handling and undoing	excellent	poor	good	good[3]	good	poor	poor	fair	fair	good
J. macro power	excellent	fair	fair	excellent	good	fair	good	good	good	fair
K. subprocesses	good	good	none	good	excellent	none	none	none	good	good
L. speed	good	good	fair	fair[4]	good	fair	good	good	poor[6]	good

Notes:
[1] On-line documentation is excellent, current written documentation inadequate.
[2] Documentation is good except for the macro language.
[3] Error handling is relatively poor, but the undo facility is excellent.
[4] On memory bound applications, quite good. But accesses the disk frequently to load various macros.
[5] With WORDIX combination, it is, of course, excellent.
[6] Its macro language is so slow as to be unusable for even simple tasks.

Table 15
Summary

source code file to disk, to invoke the compiler, and, if any errors are generated, to return to the source file and place the cursor on the offending line of code! This same procedure is used to "tokenize" programs written in BRIEF's Lisp-like macro language and load them into BRIEF.

BRIEF comes with a library of macros, including a brace checker, routines for automatic indentation of C code, incremental search, and keystroke to ASCII translation. I have used BRIEF's macro language to write a programmer's calculator, C language templates and editing routines, disk utility routines, and a program to translate one formatting language into another. I found the language easy to use and more than adequate for even complex editing needs.

With its macro and help libraries and its macro language compiler, BRIEF takes up half a double-sided IBM diskette. I have used it extensively on my two-drive Zenith without any difficulty, but it is better suited to a hard-disk system.

EC (C Source)

Except for a macro language and virtual memory buffering of files, EC offers most of the functions a programmer needs at a very affordable price. If you have sufficient RAM, you can work on five files at once, moving quickly and easily between them. The keyboard is partially reconfigurable, allowing you to create keystroke macros for twenty function key combinations and the twenty-six alphabetic control keys. There is a nicely implemented interface to DOS that enables you (once again if you have sufficient RAM) to run programs from within EC. The on-line help and tutorial (and associated menu system) were the best of the bunch.

In addition to the standard editing commands found on most editors, EC offers a number of unusual features. There is, for example, a List command that searches for a specified string and then presents you with a list of the lines containing the string together with the line number. Find, Replace, and List can be made to work over specified areas of marked text. There is, however, no backward Find or Replace and only one wild card, substituting for any single character, is supported.

EC implements a calculator that does arithmetic, logical, and shift operations and also translates characters into their ASCII codes. EC also has routines for comparing files and matching opening and closing braces, parentheses, and square brackets. If you wish, EC will automatically indent and cancel indent after opening and closing braces to facilitate the production of structured code.

EC offers no file buffering to disk, either manual or automatic, so you must have enough RAM for your files. This is even more important if you intend to run other programs from within EC.

Over the course of this review I was sent at least a half dozen updates. Each new revision had additional, useful features but also a complement of new bugs and quirks. The final review version was unable to do a backup and save on my test file or on my chart of features; I lost an hour's worth of work as a result. The authors of EC are available by phone,

and eager and willing to respond to suggestions and complaints, so I assume this problem will be quickly fixed. If they would settle down with a stable version and get all the bugs out, EC would be an excellent editor for the price.

EDIX (Emerging Technology)

EDIX is the editor in Emerging Technology's *Professional Writer's Package*. The other modules in this package are WORDIX, a text formatter, SPELLIX, a spelling checker, and INDIX, an automatic indexer. The separation of editor and formatter will be familiar to all who have used the Unix editor vi and text formatter nroff. As a text formatting system, the EDIX/WORDIX combination is powerful

Command	Function
?	Matches any single character.
*	Matches any number of occurrences of any character.
@	Matches any number of occurrences of the preceding character or expression.
|	Matches either the preceding or following character or expression (alternative).
{	Begins a group of expressions.
}	Ends a group of expressions.
\	Escape character; if the following character has a special meaning, it is treated as a normal character, and vice-versa.
\n	Matches a newline (carriage-return, line-feed) sequence.
\t	Matches a tab character.
\c	Places the cursor under the following matched character
< or %	Matches the beginning of a line.
> or $	Matches the end of a line.
\#	Substitute the actual text matched by the #th matched group of expressions (# is a digit from 0 to 9).
[. . .]	Matches any *one* of the characters between [and].
[~ . . .]	Matches if the character is not any one of the characters between [~ and].
a–b	Matches (or does not match) any character from the range a to b when within [. . .] or [~ . . .] (a and b can be any character).

Table 16a
BRIEF's Regular Expressions

Command	Function
?	Matches any single character.
*	Matches any number of occurrences of any character.
@	Matches any number of occurrences of the preceding character or expression.
Alt-N	Matches a newline (carriage-return, line-feed) sequence.
%	Matches the beginning of a line.
$	Matches the end of a line.
Alt-L	Substitute the actual text matched by the search pattern. This is the only special character that can be used in the replacement string.
[. . .]	Matches any *one* of the characters between [and].
[^. . .]	Matches if the character is not any one of the characters between [~ and].
a–b	Matches (or does not match) any character from the range a to b when within [. . .] or [^. . .] (a and b can be any character).

Table 16b
EDIX's Regular Expressions

and flexible. It has served as the kernel of such sophisticated applications as AcademicFont, a multilingual text formatter that provides output in Greek, Russian and other European languages with non-Roman characters. As a stand-alone programmer's editor (available separately), EDIX offers most of the commands that a programmer would need, along with some extras such as regular expressions in searches. It does not support virtual memory, although it can use all 640K of RAM in your machine. It can handle twelve files simultaneously with four windows on the screen. You can change directories, see directory listings, and erase files from within the editor. You can execute DOS internal commands from within EDIX, but you cannot run subprocesses such as a compiler.

EDIX supports keyboard macros. When you define a keyboard macro in EDIX, you write it into a buffer using various mnemonics for the keystroke commands. For example, to assign to Alt-F1 a macro that places C-style comments around a single line and then moves down a line, you would place in an empty buffer:

```
f 104 @aly/*@alz */@lnd
```

and then invoke the config command. If you wished permanently to assign this keystroke macro to the key combination Alt-F1, you could add this line to your initialization file and EDIX would read it in automatically at startup. Using macros such as these, you can completely reconfigure the keyboard. EDIX will not, however, support assignments involving multiple keystrokes such as WordStar's ^K^M move-block command.

EDIX allows you to move or copy lines, partial lines, and columnar data. The latter feature is particularly nice. Unfortunately, if you wish to move or copy parts of lines, you must subsequently remove the newline characters that EDIX automatically places at the beginning and end of blocks. This may be acceptable when using WORDIX, which will rejustify text lines, but is an inconvenience in moving blocks of code such as parameter lists from the middle of one line

to the middle of another. You can, however, write a macro to eliminate this problem.

EDIX is easy to learn and to use. It comes with both on-line help and tutorial. It is not very fast—probably because it is constantly updating the screen. For example, in the search and replace command that I used to strip comments from my test file, every character deletion occurred one at a time and the screen was updated accordingly!

Emacs (UniPress Software)

The original Emacs was written by Richard Stallman at MIT. Over the years it has spawned a host of imitations. Two of the editors reviewed in this article, BRIEF and Epsilon, are spiritual kin. The Emacs distributed by UniPress Software is based on James Gosling's Unix version of Emacs. It is most faithful to the full glory (and complexity) of the original Emacs that I used for years on a DEC-20. It has the capacity to handle multiple files and windows, to run scores of macro programs, to define and redefine keys and macros on the fly, to transform itself into a specialized editor for specialized chores, in short, to do just about anything you might want an editor to do. In order to have this much power, however, you must pay a price: disk accesses are frequent, a massive amount of memory and disk capacity is required and the documentation is less than complete. You should also keep in mind that a couple of the editors under review here are almost as powerful as Emacs.

The original Emacs was an extension of the TECO editor and used an extended TECO as its macro language. UniPress' Emacs uses "mock lisp" (MLisp) instead. I personally find Mlisp much easier to use. Emacs comes with a rich collection of Mlisp programs, including such delights as "electric-c", which substantially automates the structuring of C code, "abbrev," which allows you to specify abbreviations for Emacs to expand automatically as you type, and "undo", which allows you to undo the last 100 commands or 1000 characters, whichever is reached first. The search and replace commands can use regular expressions that allow complex conditional searches and translations.

You need at least 384K to run Emacs, and it really isn't comfortable with less than 512K. The program, which weighs in at 237K before allocating any of its many buffers, is totally memory resident and has no virtual memory management. The files you edit must fit into the remaining memory. Emacs also frequently accesses the disk for various MLisp programs that it needs, so, although you can run it on a two-drive floppy disk system, I would not recommend it. It performs fairly well on my Enhanced PC-AT.

The documentation I received is incomplete. A number of the macro packages included with the editor come with little or no explanation beyond the comments in the code files. Because I had become familiar with Emacs on the DEC-20, I found that I could fill in many of the gaps, but I would not want to begin my exposure to Emacs with this manual. There was mention of the "Info" data base for on-line help and a "Teach" tutorial, but neither was included. This Emacs is really closer to a pre-release or Beta

version, which its version number (0.80) also suggests. UniPress also supplies a version of its Emacs for VAX/VMS and Unix systems.

Epsilon (Lugaru Software)

Epsilon closely resembles Emacs. The one major difference is that the current release does not have a macro language. It does, however, have a full-featured keystroke macro facility, which can be used to extend the editor in useful ways. The keyboard is completely reconfigurable, and custom-made keystroke macros can be assigned to any key or given a name. You can easily edit a number of large files simultaneously and have up to eleven windows open at once. Epsilon handles the virtual memory management automatically. It offers several types of search command, including incremental search both forward and backward. It does not, however, have wild cards. It handles editing functions with more speed than many other editors.

Epsilon is the only editor under review that allows you to continue editing while running a second program concurrently. Not all programs can be run in this way, but most compilers and linkers will work fine. There is some degradation in editing speed when the two programs are running simultaneously, especially if the background program is accessing the disk with any frequency. Nevertheless, serious editing can still be done. I did encounter one significant problem with the concurrent processing. In the current version there is no way to limit the amount of memory that Epsilon uses. Instead, Epsilon takes memory as it needs it and does not release it when it is no longer being used. So, if you edit a couple of large files and use, say, the help facility, you quickly run out of memory for your second process.

Emacs, and hence Epsilon, has an especially rich set of editing commands. In a few cases, however, I wish that the the folks at Lugaru had gone beyond the original Emacs. For example, when you mark a block of text in Emacs, the only way you can see exactly what you have marked is to switch the cursor and the invisible mark back and forth. Epsilon works the same way. Other editors use reverse video or some other visual means to identify the marked block. This is a minor point, but it illustrates Epsilon's close dependence upon Emacs.

I also have had the opportunity to see a version of Epsilon in Beta test that includes a C-like macro programming language and regular expressions in searches. I was told that the final version would have regular expressions in both search and search-and-replace operations. Using the new macro language, I wrote and timed a routine to count braces. It handled my test file in 10 seconds. Using regular expressions in a search-and-replace, it took 131 seconds to strip out the comments in my test file of 500 assembly language comments. You may compare these times with other entries in Table 13, remembering that the version I examined was in Beta test.

MIX (MIX Software)

MIX is the least expensive of the edi-

Command	Function
	Matches any character except newline.
\w	Matches any word character as defined by the syntax table.
\W	Matches any non-word character.
\b	Matches at a boundary between a word and a non-word character.
\B	Matches anywhere but at a boundary between a word and a non-word character.
\¦	Matches the regular expression preceding or following it.
\`	Matches at the beginning of the buffer.
\'	Matches at the end of the buffer.
\<	Matches anywhere before dot.
\>	Matches anywhere after dot.
\=	Matches at dot.
\(. . .\)	Matches what it brackets. Used with the next command.
\n	Matches a copy of the string that the bracketed regular expression beginning with the nth \(matched.
[. . .]	Matches any *one* of the characters between [and].
[^. . .]	Matches if the character is not any one of the characters between [^ and].
*	Matches zero or more matches of the regular expression that precedes it.
^	Matches the beginning of a line.
$	Matches the end of a line.
\n	In replacement, causes the string matched by the nth bracket expression to be inserted.
&	Causes the string matched by the entire search string to be inserted.

Table 16c
EMACS Regular Expressions

Command	Function
Control-E	Match any character
Control-L	Take the next character literally. This allows a wild card character to be searched for.
Control-N	Match anything but the following character.
Control-S	Matches either a space or a blank.
Control-W	Matches any word terminator, that is, any character other than a letter or a digit.

Table 16d
Pmate's Wild Card Characters

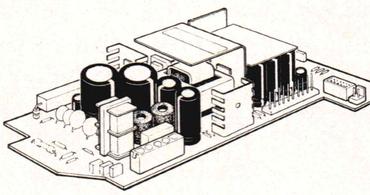
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

+ 5VDC - 8 Amps
+12VDC - 5 Amps
-12VDC - 1 Amp
(81 WATTS MAX)



$29.95 ea.

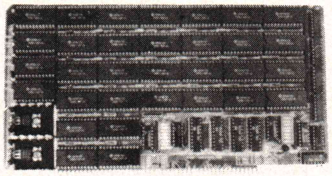
4 FOR $99

ADD $2 EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT. MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

64K S100 STATIC RAM

$119.00 KIT



LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

FOR 56K KIT $105

ASSEMBLED AND TESTED ADD $50

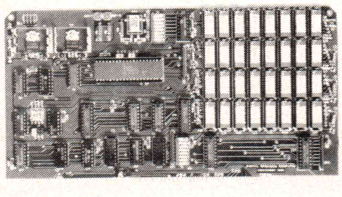
PRICE CUT!

FEATURES:
* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
* Fully supports IEEE 696 24 BIT Extended Addressing.
* 64K draws only approximately 500 MA.
* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
* SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
* 2716 EPROMs may be installed in any of top 48K.
* Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
* Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
* BOARD may be partially populated as 56K.

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



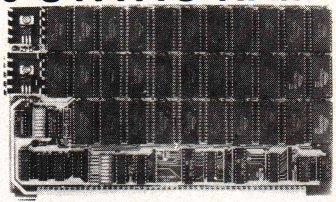
FEATURES:
* 256K on board, using + 5V 64K DRAMS.
* Uses new Intel 8203-1 LSI Memory Controller.
* Requires only 4 Dip Switch Selectable I/O Ports.
* Runs on 8080 or Z80 S100 machines.
* Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
* Provisior for Battery back-up.
* Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
* Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $69.95 (8203-1 INTEL $29.95)

(ADD $50 FOR A&T) $149.00 #LS-100 (FULL 256K KIT)

64K SS-50 STATIC RAM

$99.95 (48K KIT)



LOW POWER!

RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS $18.00

FULL SOCKET SET $15.00

56K KIT $109

64K KIT $119

ASSEMBLED AND TESTED ADD $50

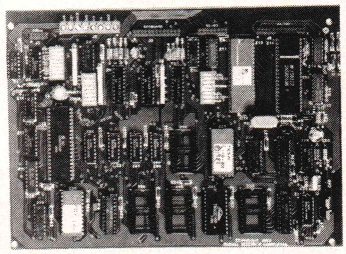
FEATURES:
* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
* Fully supports Extended Addressing.
* 64K draws only approximately 500 MA.
* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
* Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
* 2716 EPROMs may be installed anywhere on Board.
* Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
* One Board supports both RAM and EPROM.
* RAM supports 2MHZ operation at no extra charge!
* Board may be partially populated in 16K increments.

THE NEW ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:
* Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
* RS232 at 16 BAUD Rates from 75 to 19,200.
* 24 x 80 sta dard format (60 Hz).
* Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
* Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
* Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
* 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
* Composite or Split Video.
* Any polarity of video or sync.
* Inverse Video Capability.
* Small Size: 6.5 x 9 inches.
* Upper & lower case with descenders.
* 7 x 9 Character Matrix.
* Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80 (COMPLETE KIT, 2K VIDEO RAM)

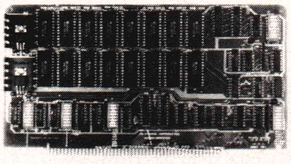
BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM $49.95

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

32K S100 EPROM/STATIC RAM

NEW! FOUR FUNCTION BOARD! NEW!



EPROM II FULL EPROM KIT $69.95

A&T EPROM ADD $35.00

BLANK PC BOARD WITH DATA $39.95

SUPPORT IC'S PLUS CAPS $16

FULL SOCKET SET $15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

PRICES SLASHED!

FEATURES:
* This one board can be used in any one of four ways:
  A. As a 32K 2716 EPROM Board
  B. As a 32K 2732 EPROM Board (Using Every Other Socket)
  C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
  D. As a 32K Static RAM Board
* Uses New 2K x 8 (TMM2016 or HM6116) RAM's
* Fully Supports IEEE 696 Buss Standard (As Proposed)
* Supports 24 Bit Extended Adressing
* 200 NS (FAST!) RAM'S are standard on the RAM Kit
* Supports both Cromemco and North Star Bank Select
* Supports Phantom
* On Board wait State Generator
* Every 2K Block may be disabled
* Addressed as two separate 16K Blocks on any 64K Boundary
* Perfect for MP/M* Systems
* RAM Kit is very low power (300 MA typical)

32K STATIC RAM KIT — $99.95

For RAM Kit A&T — Add $40

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

Circle no. 33 on reader service card.

tors. Even so, it comes with an extensive range of features: you can split the screen and edit two files at once, fully reconfigure the keyboard, and run other programs as subprocesses from within the editor. Furthermore, although MIX does not provide a full macro language, you can still create relatively sophisticated keyboard macros using combinations of the many commands built into MIX.

Unfortunately, all this power is handicapped by the small work space. MIX requires you to page larger files through memory manually. Once the working buffer is full, you must write lines out before you can load additional lines in. Other editors under review also work this way, but MIX has a particularly small buffer space. It appears to be only about 15K in size. Most commands, such as the replace command, only work over the portion of the file in the buffer. To perform a replace or to run a macro over a large file, you must read in a portion of the file, execute the commands, write out that portion, read in the next one, execute the commands, and so on. To go back to a portion already written to disk, you must write out the whole file, close it, and then read it in again. This could be done with a macro; nevertheless, it is inconvenient.

It is fairly simple to create keyboard macros to perform editing chores. The macros can be assigned to a key or combination of keys or can be given a name and executed from the command line. You can also repeat a command or macro a specified number of times. I used these two features to change all the assembly language comments in my test file to C-style comments. While doing this, I encountered several difficulties. The repeat command apparently works by placing copies of the command keystroke(s) in the keyboard buffer. This works fine as long as the number of repetitions is small, but, when that number becomes large, you run out of memory. Moreover, once the repetition of a command begins, there appears to be no way to abort, either from within a keyboard macro or from the keyboard itself. In the assembly-to-C test, I exhausted memory twice and then ended up with a series of extra */ characters at the end of the file. MIX updated the screen after every operation, which added to the time the whole process took. MIX is also available for 8-bit CP/M computers.

Command	Function
¦A	Matches any alphabetic letter, upper or lower case.
¦B	Matches a blank or tab.
¦C	Matches any control character.
¦D	Matches any numeric digit (0 through 9).
¦F	Matches any alphanumeric (letter or digit).
¦L	Matches a line terminator (carriage-return linefeed, form feed, or end of file).
¦M	Matches any sequence of zero or more characters.
¦N	Matches any character except the following character or pattern.
¦Pr	Access contents of text register r as pattern set.
¦Rr	Access contents of text register r as search string.
¦S	Matches any separator (not a letter or digit).
¦T	Matches selected separators (terminators).
¦U	Matches any upper case letter.
¦V	Matches any lower case letter.
¦W	Matches a single or multiple spaces or tabs.
¦X	Matches any character.
¦Y	Matches multiple characters until next pattern matches.

Table 16e
VEDIT PLUS's Wild Card Characters

Command	Function
Control-A	Matches any alphabetic character.
Control-B	Matches a space (ASCII 32).
Control-C	Matches a capital letter.
Control-D	Matches a digit.
Control-L	Matches a lower case letter.
Control-N	Matches any alphanumeric character (letter or digit).
Control-W	Matches a space or a tab.

Table 16f
XTC's Wild Card Characters

Command	Function
Control-Alt-A	Matches any letter or digit.
Control-Alt-L	Matches any letter.
Control-Alt-N	Matches any digit.
Control-Alt-W	Matches any arbitrary string of characters up to 80 characters in length.
Control-Alt-X	Matches any character.

Table 16g
XyWrite's Wild Card Characters

Pmate (Phoenix Computer Products)

Pmate is both lean and fast. It is an assembly language program originally written for CP/M-80 and machines with no more than 64K. Configured for a thousand byte garbage stack and a five thousand byte permanent macro area, Pmate comes to about 29K of code. Obviously, it can easily fit inside even an anemic PCjr!

Pmate operates in two modes: an

editing and a command mode. Although the editing mode is full-featured, it is in command mode that Pmate shows its muscle. Any of the more than 120 commands available in Pmate's macro language can be entered singly or as a macro program at Pmate's command line and executed immediately. Commands can also be entered into one of 10 additional text buffers and Pmate can be directed to take its commands from there. These buffers can be saved to disk, restored, and even saved inside Pmate's permanent macro area.

Pmate has a useful set of wild cards for searches (See Table 16) and a LIFO stack that holds deletions. This stack can be popped to restore deleted items back into the text. Ten macros can be placed on the ten function keys.

Pmate's macro language is powerful, but cryptic. The language is designed to be used "as you go." There is, therefore, a premium on terseness. The shorter the commands, the more quickly you can type them—provided you remember them! Most of the commands in Pmate are either one or two letters long. With only two letters, it is difficult to come up with meaningful mnemonics and, as a result, the commands are hard to remember. Compounding the problem is Pmate's documentation, which is disorganized and lacking an index. Pmate's error recovery also leaves much to be desired: it hung my machine on the disk door open test!

The preceding comments apply to Version 3.37. I also had the opportunity to use a preliminary copy of Version 4.0. This new version comes in three configurations: a menu driven version, and versions specialized for C and Fortran. Included with the package were programs that allow multitasking (Pmate and one other program running simultaneously) and support for the use of a mouse. On-line help is also available. The extensions to Pmate are based on Pmate's macro language and demonstrate its power. The menu driven and mouse versions can shield the user somewhat from Pmate's rather arcane command language. The C language version adds a number of useful macros that can speed the generation and debugging of C code. I have not programmed in Fortran for more than a decade and do not have a Fortran compiler, so I did not try the Fortran version. The other versions worked as advertised. The supplied macro-language file for the C routines is uncommented, and demonstrates how cryptic Pmate's language can be. Documentation was scant and I did encounter a few bugs, but my overall impression was positive. Version 4.0 also partially corrected the bug in error trapping; it still let DOS handle the error and the abort option returned me to the operating system, but at least the machine did not hang.

VEDIT PLUS (Compuview)

VEDIT PLUS is an enhanced version of VEDIT, an assembly language program originally written for 8-bit CP/M machines. VEDIT PLUS adds to VEDIT expanded text registers, a complete programming language, and the ability to edit multiple files. In theory, you can edit up to thirty-seven files at once. Given VEDIT PLUS' memory limitations, however, this would not be practical, unless the files were very small.

VEDIT PLUS has both an edit and a command mode. The commands in

Brief, Version 1.3
Solution Systems
335-D Washington St.
Norwell, MA 02061
(617) 659-1571 or (800) 821-2492
Price: $195.00

EC Editor, Version 2.1
C Source
12801 Frost Rd.
Kansas City, MO 64138
(816) 353-8808
Price: $49.50

Edix, Version 4.02
Emerging Technology Consultants, Inc.
4760 Walnut Street
Boulder, CO 80301
(303) 447-9495
Price: $195.00

AcademicFont
University MicroComputers
665 Monte Rosa Dr. #915
Menlo Park, CA 94025
(415) 854-8845
Price: from $275.00

Emacs, Version 0.80
UniPress Software, Inc.
2025 Lincoln Highway
Edison, NJ 08817
(201) 985-8000 or (800) 222-0550
Price: $495.00

Epsilon, Version 2.03
Lugaru Software, Ltd.
P.O. Box 110037
5227 Fifth Ave., Ste. 12
Pittsburgh, PA 15232
(412) 621-5911
Price: $195.00

MIX Editor, Version 1.2
MIX Software
2116 E. Arapaho
Suite 363
Richardson, TX 75081
(214) 783-6001 or (800) 622-4070
Price: $29.95

Pmate, Version 3.37
Phoenix Computer Products Corp.
1420 Providence Highway, Ste. 115
Norwood, MA 02062
(617) 762-5030
Price: $225.00

Vedit Plus, Version 2.01 and 2.31
Compuview
1955 Pauline Blvd.
Ann Arbor, MI 48103
(313) 996-1299
Price: $225.00

XTC, Version 3.0F
Wendin, Inc.
Box 266
Cheney, WA 99004
(509) 235-8088
Price: $99.00

Xywrite II Plus, serial number above 1 million
Xyquest
P.O. Box 372
Bedford, MA 01730
(617) 275-4439
Price: $295.00

Table 17
Product Information

Technical Product Information

FREE
For The Asking

See something you'd like to learn more about? Need more details?

DR. DOBB'S JOURNAL **Reader Service Card**

Name ______________________ Phone ____________

Address ______________________________________

City/State/Zip ________________________________

Expiration Date: Feb. 28, 1986 November 1985 #109

Please circle one for each category:

I. My firm or department is a:
A. Systems Integrator/House
B. Software Dev. Firm
C. Hardware OEM or Manuf.
D. DP, MIS or Data Service
E. Consulting Firm
F. Eng. or Science Lab.
G. Other

II. My job function is:
H. Company Mgmt./Admin.
J. Computer Systems Mgt.
K. Programmer/Technical Staff
L. Consultant
M. Engineering Mgmt. or Staff
N. Scientific Mgmt. or Staff
O. Other

III. Number of employees in my firm:
1. Less than 10
2. 10—99
3. 100—499
4. 500—9,999
5. 10,000—or More

IV. This inquiry is for:
P. Immediate Purchase
Q. Future Project

V. Purchasing Authority
(check all that apply)
R. Recommend or Specify
S. Final Decision-Maker
T. No Influence

VI. I advise others about computers, on the average:
6. More Than Once-A-Day
7. Once-A-Day
8. Once-A-Week
9. Not At All

VII. I design/write software professionally
U. Yes
V. No

VIII. I buy computer products through:
(check all that apply)
W. Retail Stores
X. Mail Order Houses
Y. On-site Direct Salespeople
Z. All of the Above

IX. I am currently a subscriber:
A. Yes
B. No

Check each advertisement for corresponding number and circle below:

001	011	021	031	041	051	061
002	012	022	032	042	052	06
003	013	023	033	043	053	06
004	014	024	034	044	054	06
005	015	025	035	045	055	06
006	016	026	036	046	056	06
007	017	027	037	047	057	06
008	018	028	038	048	058	06
009	019	029	039	049	059	06
010	020	030	040	050	060	07

071	081	091	101	111	121	13
072	082	092	102	112	122	13
073	083	093	103	113	123	13
074	084	094	104	114	124	13
075	085	095	105	115	125	13
076	086	096	106	116	126	13
077	087	097	107	117	127	13
078	088	098	108	118	128	13
079	089	099	109	119	129	13
080	090	100	110	120	130	14

				Articles		
141	151	161	171			
142	152	162	172	181	191	20
143	153	163	173	182	192	20
144	154	164	174	183	193	20
145	155	165	175	184	194	20
146	156	166	176	185	195	20
147	157	167	177	186	196	20
148	158	168	178	187	197	20
149	159	169	179	188	198	20
150	160	170	180	189	199	20
				190	200	21

☐ Please send me a one year subscription to D
Dobb's Journal at $25.00 and bill me later.

Note:

For quicker, more effective processing of your inquiry, please provide responses to questions I through IX.

Information that's
• Current
• In-depth
• Directed to you on specific products & services

FREE

Send us your POSTAGE-PAID card Today!

DR. DOBB'S JOURNAL **Reader Service Card**

Name ______________________ Phone ____________

Address ______________________________________

City/State/Zip ________________________________

Expiration Date: Feb. 28, 1986 November 1985 #109

Please circle one for each category:

I. My firm or department is a:
A. Systems Integrator/House
B. Software Dev. Firm
C. Hardware OEM or Manuf.
D. DP, MIS or Data Service
E. Consulting Firm
F. Eng. or Science Lab.
G. Other

II. My job function is:
H. Company Mgmt./Admin.
J. Computer Systems Mgt.
K. Programmer/Technical Staff
L. Consultant
M. Engineering Mgmt. or Staff
N. Scientific Mgmt. or Staff
O. Other

III. Number of employees in my firm:
1. Less than 10
2. 10—99
3. 100—499
4. 500—9,999
5. 10,000—or More

IV. This inquiry is for:
P. Immediate Purchase
Q. Future Project

V. Purchasing Authority
(check all that apply)
R. Recommend or Specify
S. Final Decision-Maker
T. No Influence

VI. I advise others about computers, on the average:
6. More Than Once-A-Day
7. Once-A-Day
8. Once-A-Week
9. Not At All

VII. I design/write software professionally
U. Yes
V. No

VIII. I buy computer products through:
(check all that apply)
W. Retail Stores
X. Mail Order Houses
Y. On-site Direct Salespeople
Z. All of the Above

IX. I am currently a subscriber:
A. Yes
B. No

**Check each advertisement for corr-
sponding number and circle below:**

001	011	021	031	041	051	0
002	012	022	032	042	052	0
003	013	023	033	043	053	0
004	014	024	034	044	054	0
005	015	025	035	045	055	0
006	016	026	036	046	056	0
007	017	027	037	047	057	0
008	018	028	038	048	058	0
009	019	029	039	049	059	0
010	020	030	040	050	060	0

071	081	091	101	111	121
072	082	092	102	112	122
073	083	093	103	113	123
074	084	094	104	114	124
075	085	095	105	115	125
076	086	096	106	116	126
077	087	097	107	117	127
078	088	098	108	118	128
079	089	099	109	119	129
080	090	100	110	120	130

				Articles	
141	151	161	171		
142	152	162	172	181	191
143	153	163	173	182	192
144	154	164	174	183	193
145	155	165	175	184	194
146	156	166	176	185	195
147	157	167	177	186	196
148	158	168	178	187	197
149	159	169	179	188	198
150	160	170	180	189	199
				190	200

☐ Please send me a one year subscription t
Dobb's Journal at $25.00 and bill me late

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal
P.O. BOX 13851
PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT #27346, PHILADELPHIA, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE TOOLS FOR ADVANCED PROGRAMMERS
Dr. Dobb's Journal
P.O. BOX 13851
PHILADELPHIA, PA 19101

DESMET C $150*

Macintosh tm Development Package

Runs on both**128K** and **512K Macintosh**
Full **K&R** Compiler — **IEEE** Floating Point
>450 Function Macintosh **ROM** Library
Assembler, Linker & Librarian
>120 Function STDIO Library
Machine Code Debugger
Source Code Editor
"Shell" interface
300 Page manual
RAM Disk

Published and sold direct to the end-user by:

C Ware Corporation
P.O. Box C, Sunnyvale, CA 94087
(408) 720-9696

*Price includes domestic shipping. Canada $5. Europe/Asia add $20. Call to **charge by VISA, MC or AMEX.** Street address: 505 W. Olive, #767, Sunnyvale, CA 94086.

Macintosh is a trademark licensed to Apple Computer, Inc.

Circle no. **57** on reader service card.

Release 2.0

C-INDEX™

Variable Length Record Management For C

C-INDEX is a state-of-the-art data management function library for C programmers. Ideal for all data and text based applications. No other package can give you the performance, capability, and portability of C-INDEX. To make sure you are a satisfied customer, we offer a 30 day money-back guarantee. Ask us about it.

- variable length data storage with B+Tree indexing
- high performance, easy to use
- large and small models, fully transportable source
- IBM PC format: Lattice, Microsoft, C86, others
- Macintosh format: Consulair, Manx, others

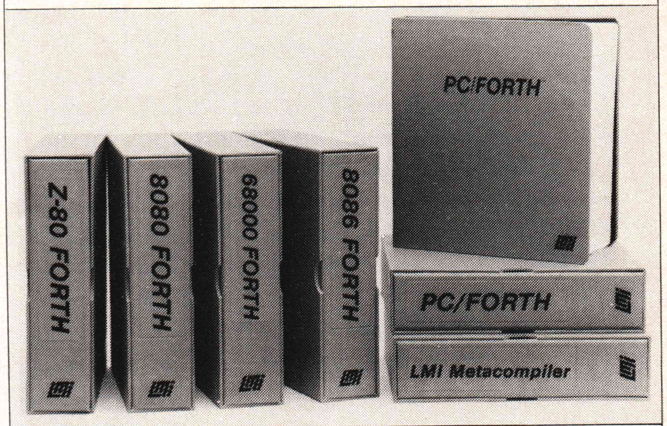
C-INDEX/FILE ($99)	Object code package.
C-INDEX/PRO ($195)	Partial source code, no royalties.
C-INDEX/PLUS ($395)	Complete transportable source code, no royalties.

Trio Systems

2210 Wilshire Blvd., Suite 289
Santa Monica, CA 90403
213/394-0796

Circle no. **150** on reader service card.

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, and 6502
- No license fee or royalty for compiled applications

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, D-7820 Titisee-Neustadt
UK: System Science Ltd., London EC1A 9JX
France: Micro-Sigma S.A.R.L., 75008 Paris
Japan: Southern Pacific Ltd., Yokohama 220
Australia: Wave-onic Associates, 61,07 Wilson, W.A.

Circle no. **55** on reader service card.

the edit mode can be assigned to whatever keys you wish, with one important limitation: VEDIT PLUS allows you to designate only two prefix keys. Many editors use prefix keys. For example, in the WordStar command ^Q^D, which means go to the end of the current line, ^Q is a prefix key. Another frequently used prefix key in WordStar is ^K. If you were to reconfigure your keyboard with VEDIT PLUS to resemble the WordStar keyboard, you could designate only two prefix keys, say, ^Q and ^K. In that case, you could not use the function and cursor keys because they also require a prefix key.

VEDIT PLUS provides a full TECO-like macro language. You can use this language to write editing routines either on the command line or in a text register from which the editor can be directed to take its commands. These macro programs can be edited, saved to disk, and restored for later use. In the current version of VEDIT PLUS these macros cannot, however, be assigned to keys. VEDIT PLUS also offers a particularly robust set of wild cards that approach the functionality of regular expressions, at least for searches (See Table 16). You can even search for alternate patterns; that is, search, say, for "int" or "char" at the same time. You cannot, however, search backward. When you combine the macro language with the wild cards, you can do sophisticated translations. To give you an idea of the language's potential, there is a separate product from CompuView, a macro program, that allows VEDIT PLUS to translate Z80 assembly code to 8086 code. With VEDIT PLUS itself comes a file comparison macro, a mailing list sorter, and an on-line calculator.

Like Pmate's macro language, VEDIT PLUS' language is terse and somewhat cryptic. As I mentioned before, it is impossible with only one or two letters to give meaningful names to as many commands and internal variables as a true macro language offers. VEDIT PLUS' macro language is, fortunately, reasonably well documented, although the macro language manual needs an index.

The current version of VEDIT PLUS still lacks a number of features found in other editors. Although you can use automatic disk buffering as you move forward and backward in a large file, there is no virtual memory *per se* and the total memory reserved for all text and macro buffers is a restricting 64K. You cannot view two files at one time, but must switch between them. You also cannot run a program from within VEDIT PLUS, and DOS 2.0 pathnames are not yet supported.

I also previewed a version of VEDIT PLUS in Beta test that adds brief on-line help, pathname and subdirectory support, and reverse search. In the Beta version each buffer can allocate up to 60K of RAM, a significant improvement over the older version. I am told that additional enhancements, including true virtual memory management, are planned.

XTC (Wendin, Inc.)

XTC is the only editor of the nine to have its source code included! If you wish, you can can change the code itself and recompile the editor. The code is also a fine source for useful Pascal, C, and assembler routines.

The editor itself is full-featured and powerful, boasting its own macro language and the ability to do multi-

tasking within the editor itself. This latter feature is unique among the nine editors. This is the ability to run a macro program in the background and continue with editing in the foreground. In Version 3.0, which is a substantial upgrade of Version 2.0, you can invoke DOS and run compilers and other programs from within the editor. This version also allows wild cards in search strings (see Table 16) and auto-indention for C or Pascal programming. Searches proceed only forward, from the current cursor location. The keyboard cannot be reconfigured.

The editor is line-oriented in many of its commands. For example, with XTC's block move commands you move whole lines around rather than portions of lines. Similarly, the Undo command restores only deleted lines rather than characters or words deleted within a line. This limitation prevents you from, for example, copying just the parameter list of a procedure from one location to another; instead you must copy the whole line.

The macro language includes the full range of loops and conditional structures and an extensive collection of functions that allow you to keep track of the editor environment. Macros can either be entered in their keystroke form and used immediately, or written out in a longer form and compiled with a separate macro compiler. They are limited, however, to eighty keystrokes each, so even fairly simple macros must be broken into several pieces. Macros can be saved by name or assigned to function keys. In branching tests only one Boolean operator is allowed. In other words, you cannot evaluate a compound Boolean expression in one test. It is this limitation, and the fact that the program is constantly updating the screen during macro programs, that probably accounts for XTC's incredibly slow performance on some of the benchmarks. Whereas, for example, BRIEF took 6 seconds to count all the curly braces in my test file and VEDIT PLUS, which had the next slowest time, took 5 minutes and 36 seconds, XTC took over 54 minutes! Until Wendin substantially speeds up its macro language, it is, for all practical purposes, unusable.

XyWrite II Plus (Xyquest)

XyWrite II Plus is sold primarily as a word processor, and as such has garnered some outstanding reviews[2]. But it can also be used successfully as a programming, and programmable, editor, thanks to its reconfigurable keyboard, keystroke programs, and its macro programming language. It also allows you to run a compiler from within the editor.

Written in assembler, XyWrite is fast and compact, weighing in at only 72K, yet offers all the features of a powerful "what you see is what you get" (WYSIWYG) word processor. It boasts a useful set of wild cards for searches (see Table 16). The last deletion can be restored, be it word, line, or block. To jump to a particular line in a file, you must calculate the page on which the line would reside.

Using XyWrite's extensive list of functions, you can produce editing programs with conditional branching. These programs can be saved to disk, reloaded, and executed from XyWrite's command line. Unfortunately, the programming language lacks certain crucial commands (such as a command to determine the character on which the cursor rests) and is scantily documented. XyQuest explicitly states that it will not provide phone support for the features of its macro language.

XyWrite takes over the keyboard, so it cannot be run with SuperKey and some other memory resident programs. It will work with SideKick, if you are careful to follow certain rules about opening and closing SideKick's windows.

Notes

1 For an MSDOS version of grep, see Allen Holub's article "grep.c: A Unix-like Generalized Regular Expression Parser" in *DDJ*, October 1984, #96.

2 See, for example, "Xywrite: Way to Go" by Peter H. Weil in *PC Magazine*, June 12, 1984.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 196.**

Reviews Listing (Text begins on page 60)

```
;**
;**   BRIEF -- demonstration program: brace counter
;**
(macro bracecnt
    (
        (int  count
              cur_char
        )
        (string  msg_pattern
                 msg_text
        )
        (= msg_pattern "Excess { count = %d")
        (top_of_buffer)
        (while (search_fwd "[\\{\\}]")              ; look for { or }
            (
                (= cur_char (index "{}" (read 1)))  ; determine which you found
                (if (== cur_char 1)                 ; IF cur_char == 1 THEN
                    (++ count)
```

(Continued on next page)

```
                ; ELSE
                (-- count)
            )
            (next_char)
        )
    )
    (sprintf msg_text msg_pattern count)
    (message msg_text)
  )
)
```

```
; **
; ** EMACS -- demonstration program: brace counter
; **
(defun
    (bracecnt count
        (beginning-of-file)
        (while (! (eobp))                        ; while not at buffer end
            (re-search-forward "{\\|}")          ; find either { or }
            (if (= (preceding-char) '{')         ; IF == { THEN
                (++ count)
                ; ELSE
                (-- count)
            )
            (forward-character)
        )
        (message (concat "Excess { count = " count))
    )
)
```

```
;**
;**  PMATE -- demonstration program: brace counter
;**
ØVl               ; set variable l to Ø
ua                ; go to beginning of file
[                 ; begin loop
   @CV2           ; put character number in variable 2
   @t="{          ; if char == {
      [VAl]       ;       increment variable l
   @t="}          ; if char == }
      [-lVAl]     ;       decrement variable l
   m              ; move forward one character
@C=@2             ; until at end of buffer
]                 ; end loop
                  ; Since there is apparently no easy way to display
                  ; the number on the status line
b2e               ; go to buffer 2
@l\               ; insert the result into buffer 2
```

```
R*
R*     VEDIT+ -- demonstration program: brace counter
R*
B                 ! go to the top of the register !
ØXSl              ! set numeric register l to Ø!
[                 ! begin loop

                  ! IF char == { THEN increment variable l !
(.c = 123)[
lXAl
]
```

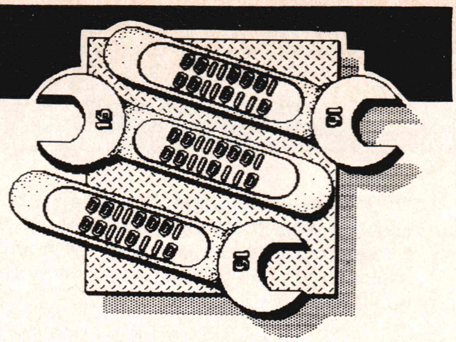
(Continued on page 86)

```
                ! IF char == } THEN decrement variable 1 !
(.c = 125)[
-1XA1
]
c               ! move a character !
(.c = 26)JL           ! IF end of file exit loop !
]               ! end loop
@YT/Excess { = /! type result!
XT1             ! stored in numeric register 1 !
```

```
--
-- XTC demonstration program: brace counter
--

macro "braces" is
   i1 := 0;                -- integer variable 1 is set to 0
   bottom_of_file;
   i2 := LN;               -- number of last line, used to check if done
   top_of_file;            -- go to beginning
   BRACES2;                -- call BRACES2
end macro;


macro "IN_LINE" is         -- must be all caps to work
   goto_column (1);        -- go to beginning of line
   repeat loop
      if CC = 123 then     -- if current character is {
         i1 := i1 + 1;
      end if;
      if CC = 125 then     -- if current character is }
         i1 := i1 - 1;
      end if;
      right_character;
   until CN > LC
   end loop;               -- until past last column with non-blank character
end macro;

Macro "BRACES2" is
   IN_LINE;                -- do first line
   repeat loop
      down_line;
      IN_LINE;
   until i2 = LN
   end loop;              -- until on bottom line of file
   display_variable (i1);   -- print out the result
end macro;
```

End Listing

16-BIT SOFTWARE TOOLBOX



by Ray Duncan

The programs published in this month's column are available for downloading from the Laboratory Microsystems RBBS *at (213) 306-3530 (300 baud or 1200 bps).*

68000 Square Roots

Mike Morton of Boston writes: "I was fascinated by the speed and elegance of Jim Cathey's 68000 square root routine (*DDJ* May 1985). If Jim is really concerned about speed he might consider these simple changes; perhaps your readers would be interested in them too. . . .

"1. Small change: the move.w #15,D4 can be changed to moveq #15,D4, assuming you don't care about the upper half of D4 (dbra is a work-oriented loop). This saves four cycles—big deal.

"2. Never shift when you can add: change each asl.l #1,Dn to add.l Dn,Dn. This reduces a ten-cycle instruction to six, saving four cycles. There are three such instructions in each loop, each executed sixteen times. Total savings, $4 \times 3 \times 16 = 192$ cycles.

"3. Let's get desperate: surprisingly, you can trade in the roxl.l #1,Dn instructions for addx.l Dn,Dn (I haven't tested this). This saves two cycles in two different places in the loop, for a total of 64 cycles.

"The total savings are 260 cycles. This is an improvement of 14% to 17% over Jim's times. Suppose we apply the changes to the unrolled word-version of the loop that Jim suggests. The only change is for technique 3, in which roxl.w #1,Dn turns into addx.l Dn,Dn 15 times. Each of these saves four, not two, cycles—this means the savings are 2×15 cycles more than in the longword version. The total difference from the original word-size routine to the new one is 290 cycles, almost a 50% improvement."

For those who missed it, Jim Cathey's original code is run in this issue as Listing One (page 90).

68000 Binary Search & PRNG

Dr. Michael P. McLaughlin of McLean, Virginia, writes: "I am submitting the two following programming 'quickies' in response to your repeated call for 68000 code. I was a bit hesitant to send them in because I suspected they might be a bit too trivial for your typical reader. Nevertheless, I have used them extensively in studies in machine learning and am certain of their usefulness.

"The first [Listing Two, page 90] is a slightly modified binary search routine. The modification [consists in the fact that the search routine returns] the place where the missing item would be. This enables one to insert the missing item and keep the list in order. A short example of the proper use of the search routine is also included [Listing Three, page 90].

"The second routine [Listing Four, page 91] was prompted by Dave Cortesi's discussion of pseudo-random number generators [PRNGs (see the February and October '85 issues of *DDJ*)]. Because this is a well-studied field there is no reason to be uncertain of the quality of the generator. The one here is statistically very good, with a very long period. It could have been written in straightforward double precision, but that code, albeit 50% shorter, requires more time per random deviate."

80286 vs 8086

Ross P. Nelson of San Jose, California, writes: "Someone recently showed me a copy of your column in which the relative performance of the Intel 80286 was being discussed [see the April '85 issue of *DDJ*]. As a former Intel employee who has worked directly with the 286 for some time, I hope I can clear up some of the confusion.

"Let me begin by comparing the 8088 and 8086. They run at approximately the same speed when performing register-to-register operations. When memory references are included, however, the 8088 requires an extra four clocks for each 16-bit memory operation. This translates to a 20% to 30% performance degradation in a common instruction mix.

"When comparing the 80286 (real mode) to the 8086 you see about a 150% performance increase in arithmetic register-to-register operations. Memory reference performance varies, depending on how the operand is specified. I've enclosed a Table [page 89] that shows the instruction timing for different memory reference instructions. The base time is the time to execute the instruction only, the EA time is the setup time required to compute the Effective Address. All times are listed in clocks.

"As is shown in the table, memory reference instructions on the 286 show a 200% to 500% performance improvement when compared to their 8086 counterparts. For an average instruction mix, therefore, it is reasonable to claim that a 286 will run about twice as fast as an 8086 operating at the same clock rate and about 2.5 to 3 times as fast as an 8088.

"The standard instruction timings shown in the table are not affected by placing the 286 into Protected Mode. In Protected Mode, however, the 286 takes a substantial performance hit every time a segment register is loaded. For example, the instruction

MOV DS, AX

which requires 2 cycles on the 8088, 8086, and 80286 in Real Mode, requires 17 cycles on the 80286 in Protected Mode. Similarly, the instruction

INT 21H

which requires 71 cycles on the 8088, 51 cycles on the 8086, and 24 cycles on the 80286 in Real Mode, takes 79 cycles on the 80286 in Protected Mode. The INT 21H figures assume an operating system call traps through a call gate to a higher privilege level."

This Month's Filter: DUMP

Richard G. Markley of La Crescenta, California, contributed the MSDOS filter that accompanies this month's column as Listing Five (page 92). It transforms the Standard Input stream into a hex and ASCII dump and writes it to the Standard Output; both input and output can be redirected. You can therefore use this filter to dump a file or character device input stream in object format to another file or to a character device such as the printer. Here are some examples:

DUMP <file
This gives a continuous scrolling dump
DUMP <file >PRN
This dumps a file on the printer
DUMP <file1 >file2
This dumps a file into another file

Pagination can be achieved with the MORE filter that is supplied with MSDOS. Here is an example:

DUMP <file | MORE

Richard writes, "MORE.COM doesn't give a consistent display. The first display has 24 lines of new text and subsequent displays have 23. This gives an unbalanced appearance when used in conjunction with DUMP. I have found the problem can be solved by this sequence of actions:

A >DEBUG MORE.COM
Use DEBUG.COM to load MORE.COM into memory.
-E 1DB 00
Replace the byte at offset 1DBH with 00H. This resets MORE's row counter to zero instead of 1.
-W
Write the change to disk.
-Q
Exit from DEBUG.

"The above patch has worked successfully on MORE.COM of DOS Versions 2.0 through 3.0."

(Listings begin on next page)

DDJ
Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 194**.

Instruction	8086 Base	8086 EA	8086 Total	80286 Base	80286 EA	80286 Total
MOV BX,[SI]	8	5	13	3	0	3
MOV CX,[1A22]	8	6	14	3	0	3
MOV AX,[BP+4]	8	9	17	3	0	3
MOV SI,[BP][DI+4]	8	11	19	3	1	4
ADD AX,[1A22]	9	6	15	7	0	7
SUB BX,[BP+2]	9	9	18	7	0	7
JMP 3A0	15	0	15	8	0	8
JMP [73EE]	15	6	21	8	0	8

Figure 1
Ross Nelson's comparison of some instruction times on the 8086, and 80286.

16-Bit (Text begins on page 88)

Listing One

```
*   Integer Square Root (32 to 16 Bit)
*
*   (Exact method, not approximate)
*
*   Call with:
*           D0.L = Unsigned number
*
*   Returns:
*           D0.L = SQRT (D0.L)
*           D1.L <> 0 if not exact root
*
*   Uses:
*           D1-D4 as temporaries ---
*           D1 = Error term
*           D2 = Running estimate
*           D3 = High bracket
*           D4 = Loop counter
*
*   Notes:
*           Result first in D0.W, but is valid in longword.
*           Takes from 1480 to 1832 cycles (including RTS).
*           (Word version is from 548 to 660 cycles).
*
lsqrt       move.w      #15,d4      ; Loop count (bits-1 of result)
            moveq       #0,d1       ; Result in D1
            moveq       #0,d2
sqrtl       asl.l       #1,d0       ; Get 2 leading bits at a time and
            roxl.l      #1,d1       ;  into Error term for extrapolation.
            asl.l       #1,d0       ; (Classical method, easy in binary)
            roxl.l      #1,d1
            asl.l       #1,d2       ; Running estimate * 2
            move.l      d2,d3
            asl.l       #1,d3
            cmp.l       d3,d1
            bls         sqrt2       ; New error term > 2* running est.?
            addq.l      #1,d2       ; Yes, we want 1 bit then.
            addq.l      #1,d3       ; Fix up new error term.
            sub.l       d3,d1
sqrt2       dbra        d4,sqrtl    ; Do all 16 bit-pairs.
            movel       d2,d0       ; Returns answer in D0.W
            rts
```

End Listing One

Listing Two

```
;BINARY SEARCH -- (USES D4-D7, A6)
;TO SEARCH A SORTED ARRAY OF SIGNED LONGWORDS BEGINNING WITH A DUMMY ENTRY
;SMALLER THAN ANY POTENTIAL ENTRY, INITIALIZE THE REGISTERS AS FOLLOWS:
;       A6 = BASE ADDRESS OF ENTIRE ARRAY (LONGWORD)
;       D4 = TARGET LONGWORD
;       D7 = LENGTH OF ARRAY, IN BYTES, LESS DUMMY (LONGWORD)
;THE SEARCH WILL RETURN, IN D6, THE DISPLACEMENT (NUMBER OF BYTES FROM BASE)
;OF THE START OF THE TARGET LONGWORD, IF PRESENT, OR -DISPLACEMENT IF THE
;TARGET IS ABSENT.
;
;
                ENTRY       BINSRCH
BINSRCH         CLR.L       D5                  ;D5 = pointer to bottom
                CLR.L       D6
B1              CMP.L       D5,D7               ;bottom > top ?
                BMI.S       FAILURE             ;yes, exit
                MOVE.L      D7,D6               ;else D6 = (D5+D7) div 2
                ADD.L       D5,D6
                LSR.L       #1,D6
                AND.L       #0FFFFFFFDH,D6      ;back to longword boundary
                CMP.L       0(A6,D6.L),D4       ;is this it?
                BEQ.S       SUCCESS             ;yes
                BGT.S       B2                  ;no, target is bigger
                SUBQ.L      #4,D6               ;target is smaller
                MOVE.L      D6,D7               ;try lower half
                BRA         B1
B2              ADDQ.L      #4,D6               ;try upper half
                MOVE.L      D6,D5
                BRA         B1
FAILURE         NEG.L       D5                  ;return -displacement
                MOVE.L      D5,D6
SUCCESS         RTS
                END         BINSRCH
```

End Listing Two

Listing Three

```
;EXAMPLE -- WHEN APPROPRIATELY LINKED, THIS CALLING ROUTINE WILL GIVE THE
;RESULTS INDICATED.
;
;
                EXTERN      BINSRCH
EXAMPLE         LEA         ARRAY,A6
```

```
          MOVE.L    #-128,D4          ;would be entry #1
          MOVE.L    #28,D7
          BSR       BINSRCH           ;returns -4
          MOVE.L    #-14,D4           ;entry #1
          MOVE.L    #28,D7
          BSR       BINSRCH           ;returns 4
          MOVE.L    #492,D4           ;entry #3
          MOVE.L    #28,D7
          BSR       BINSRCH           ;returns 12
          MOVE.L    #987654321,D4     ;entry #7
          MOVE.L    #28,D7
          BSR       BINSRCH           ;returns 28
          MOVE.L    #1000000000,D4    ;would be entry #8
          MOVE.L    #28,D7
          BSR       BINSRCH           ;returns -32
          RTS
ARRAY     DC.L      -99999            ;dummy
          DC.L      -14,27,492,10768,10112255,30741234,987654321
          END       EXAMPLE
```

End Listing Three

Listing Four

```
;PSEUDO-RANDOM NUMBER GENERATOR -- (USES D3-D7)
;GIVEN ANY SEED (1 TO 2**31-2) IN D7 (LONGWORD), THIS GENERATOR YIELDS A NON
;REPEATING SEQUENCE (RAND(I)) USING ALL INTEGERS IN THE RANGE 1 TO 2**31-2.
;THE AVERAGE EXECUTION TIME IS 342 MICROSECONDS (AT 8 MHz). THIS GENERATOR,
;REFERRED TO IN THE LITERATURE AS "GGUBS," IS KNOWN TO POSSESS GOOD
;STATISTICS. THE ALGORITHM IS:
;
;       RAND(I+1) = (16807*RAND(I)) MOD (2**31-1)
;
;WHEN PROPERLY CODED, THIS ALGORITHM WILL TRANSFORM RAND(0) = 1 INTO
;RAND(1000) =522329230. THE FOLLOWING IMPLEMENTATION USES SYNTHETIC DIVISION,
;VIZ.,
;
;       K1 = RAND(I) DIV 127773
;       RAND(I+1) = 16807*(RAND(I)-K1*127773)-K1*2836
;       IF RAND(I+1) < 0 THEN RAND(I+1) = RAND(I+1) + 2147483647
;
;REFERENCE:
;       BRATLEY, P., B.L. FOX and L.E. SCHRAGE, "A GUIDE TO SIMULATION"
;       (SPRINGER-VERLAG, 1983).
;
;
RANDOM    MOVE.L    D7,D6             ;copy RAND(I)
          BSR.S     DIV               ;divide D6 by 127773
          MOVE.L    D4,D5             ;copy K1
          MULS      #-2836,D5         ;D5 = -2836*K1
          MULU      #42591,D4         ;multiply D4 by 127773
          MOVE.L    D4,D6
          LSL.L     #1,D4
          ADD.L     D6,D4
          SUB.L     D4,D7             ;D7 = RAND(I)-K1*127773
          MOVE      #4,D4             ;counter
RAN1      MOVE.L    D7,D6             ;multiply D7 by 16807
          LSL.L     #3,D7
          SUB.L     D6,D7
          DBRA      D4,RAN1
          ADD.L     D5,D7             ;D7 = RAND(I+1)
          BPL.S     EXIT
          ADD.L     #2147483647,D7    ;normalize negative result
EXIT      RTS                         ;D7 = RAND(I+1)
;RAND(I)  (31 BITS)  DIV 127773  (17 BITS)
DIV       LSL.L     #1,D6             ;shift out unused bit

          CLR.L     D4                ;quotient
          MOVE      #14,D3            ;counter
          MOVE      D6,D5             save low word of RAND(I)
          SWAP      D6
          AND.L     #0FFFFH,D6        D6 = RAND(I) DIV 2**15
DIV1      LSL       #1,D4             ;line up quotient
          LSL.L     #1,D6             ;and dividend
          LSL       #1,D5             ;shift in bit of low word
          BCC.S     DIV2
          ADDQ.L    #1,D6
DIV2      CMP.L     #127773,D6        ;trial subtraction
          BMI.S     DIV3
          SUB.L     #127773,D6        ;real subtraction
          ADDQ      #1,D4             ;put 1 in quotient
DIV3      DBRA      D3,DIV1           ;decrement counter and loop
          RTS
          END       RANDOM
```

End Listing Four

(Listing Five begins on next page)

Listing Five

```
                          page    55,132
                          title   'DUMP --- Hex and ASCII Dump Filter'
                  ;
                  ; DUMP --- filter to transform the Standard Input stream into a
                  ;         Hex and ASCII dump, which is written to the Standard Output.
                  ;
                  ; Version 1.0  Richard G. Markley  July 20, 1985

 = 000D           cr            equ     0dh             ;ASCII carriage return
 = 000A           lf            equ     0ah             ;ASCII line feed

                  ; DOS 2.x predefined handles

 = 0000           Std_Input     equ     0               ;Standard Input device or file
 = 0001           Std_Output    equ     1               ;Standard Output device or file
 = 0002           Std_Err       equ     2               ;Standard Error device or file

                  ; DOS function numbers

 = 0030           Get_Version   equ     030h            ;get current DOS version
 = 003F           Device_Input  equ     03fh            ;read from file or device
 = 0040           Device_Output equ     040h            ;write to file or device
 = 004C           Exit          equ     04ch            ;exit with return code

 0000             code          segment para public 'CODE'

                                assume  cs:code,ds:code,es:code,ss:code

 0100                           org     100h

```

```
0100                            dump            proc    far

0100  E9 021F R                                 jmp     start

                                ; data area

0103  0D 0A                     Column_Guide    db      cr,lf
0105     0B [                                   db      11 dup (' ')
               20
                    ]

0110  30 20 20 31 20 20                         db      '0  1  2  3  4  5  6  7  8  9  '
      32 20 20 33 20 20
      34 20 20 35 20 20
      36 20 20 37 20 20
      38 20 20 39 20 20
012E  41 20 20 42 20 20                         db      'A  B  C  D  E  F',cr,lf
      43 20 20 44 20 20
      45 20 20 46 0D 0A
= 003D                          Col_Guide_Size  equ     $-Column_Guide

0140     03 [                   Data_String     db      3 dup (' ')
               20
                    ]

0143  30 30 30 30 30 30         Byte_Counter    db      '000000  '
      20 20
014B     19 [                   Hex_String      dw      25 dup (?)
               ????
                    ]

017D     08 [                   ASCII_String    dw      8 dup (?)
               ????
                    ]
```

(Continued on next page)

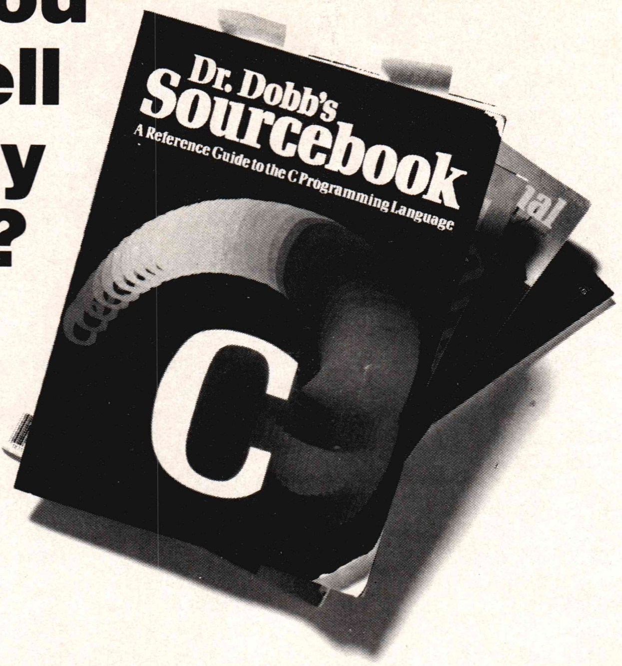
Listing Five

```
018D  0D 0A                               db     cr,lf
= 004F                  String_Size       equ    $-Data_String
= 0021                  Appx_String_Len   equu   (LENGTH Hex_String)+(LENGTH ASCII_String)

018F  0D 0A 0D 0A       New_Lines         db     cr,lf,cr,lf      ;two blank lines
= 0004                  New_Lines_Size    equ    $-New_Lines

0193  ??                Line_Count        db     ?                ;Count of lines per block

                        ; error messages

0194  0D 0A 44 55 4D 50 Pre_DOS_Error     db     cr,lf,'DUMP: incorrect DOS version',cr,lf
      3A 20 69 6E 63 6F
      72 72 65 63 74 20
      44 4F 53 20 76 65
      72 73 69 6F 6E 0D
      0A
= 001F                  Pre_DOS_Size      equ    $-Pre_DOS_Error

01B3  0D 0A 44 55 4D 50 Input_Error       db     cr,lf,'DUMP: input device error',cr,lf
      3A 20 69 6E 70 75
      74 20 64 65 76 69
      63 65 20 65 72 72
      6F 72 0D 0A
= 001C                  Input_Size        equ    $-Input_Error

01CF  0D 0A 44 55 4D 50 Empty_Error       db     cr,lf,'DUMP: missing input error',cr,lf
```

```
        3A 20 6D 69 73 73
        69 6E 67 20 69 6E
        70 75 74 20 65 72
        72 6F 72 0D 0A
 = 001D                           Empty_Size      equ     $-Empty_Error

 01EC   0D 0A 44 55 4D 50         Output_Error    db      cr,lf,'DUMP: output device error',cr,lf
        3A 20 6F 75 74 70
        75 74 20 64 65 76
        69 63 65 20 65 72
        72 6F 72 0D 0A
 = 001D                           Output_Size     equ     $-Output_Error

 0209   0D 0A 44 55 4D 50         Disk_Full_Error db      cr,lf,'DUMP: disk is full',cr,lf
        3A 20 64 69 73 6B
        20 69 73 20 66 75
        6C 6C 0D 0A
 = 0016                           Disk_Full_Size  equ     $-Disk_Full_Error


 021F   B4 30                     Start:          mov     ah,Get_Version  ;check version of DOS
 0221   CD 21                                     int     21h

 0223   0A C0                                     or      al,al           ;is it DOS v. 2.0 or higher?
 0225   75 03                                     jnz     Prepare         ;yes, proceed
 0227   E9 02E0 R                                 jmp     Error_1         ;no, output error message

 022A                             Prepare:                                ;prepare for processing
 022A   FC                                        cld
 022B   BD FFFF                                   mov     bp,-1           ;prevent output if no input
 022E   E8 0324 R                                 call    Input           ;perform input of data

 0231                             Main_Loop:                              ;output heading
 0231   B9 003D                                   mov     cx,Col_Guide_Size
 0234   BA 0103 R                                 mov     dx,offset Column_Guide
```

(Continued on page 97)

Who Says You Can't Tell A Book By Its Cover?



Dr. Dobb's Sourcebook: A Reference Guide for the C Programming Language

For years, serious programmers have relied on Dr. Dobb's Journal for the technical tools of their trade. Now, Dr. Dobb's presents the definitive programmers guide to the who, what, where, when and why of C, the leading language among software developers. This comprehensive guide to new information, products and services specific to C will be your most often-used reference!

In this valuable guide you'll find:

- An extensive directory of hardware and software services—including classes and seminars, C programming opportunities, and on-line services
- A bibliography with over 300 listings of available articles and books on C
- A comprehensive C product listing—including C compilers, graphics modules, utilities, editors and development systems, and more!
- And much more practical C programming information

At only $7.95, no C programmer can afford to be without this unique reference.

To order by credit card, call toll free: 1-800-528-6050 ext.4001 . Ask for item 004 or mail this coupon, along with payment to **Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303**

PAYMENT MUST ACCOMPANY YOUR ORDER

_____ I enclose check/money order

_____ Please charge my ___VISA ___M/C ___ American Express

Card #_______________________________ Exp. Date _____

Signature ___________________________________

Name ___________________________________

Address ___________________________________

City ____________________ State ______ Zip ______

Please allow 6 to 12 weeks for delivery

Please send me _______ copies of **Dr. Dobb's Sourcebook**

at $7.95 each = ____________

+ Shipping & Handling = ____________

(Must be included with order. Please add $1.50 per book in U.S. $3.25 each surface mail outside U.S. Foreign airmail rates available on request.)

TOTAL = ____________

3109C

Listing Five

```
0237  E8 0341 R                    call    Output
                                                        ;now dump a block
023A  C6 06 0193 R 08              mov     Line_Count,8    ;number of lines per block
023F  EB 0C 90                     jmp     Do_Block
0242                Output_Lines:                       ;send 2 blank lines
0242  B9 0004                      mov     cx,New_Lines_Size
0245  BA 018F R                    mov     dx,offset New_Lines
0248  E8 0341 R                    call    Output
024B  EB E4                        jmp     Main_Loop
024D                Do_Block:                           ;initialize part of data
                                                        ;template with spaces
024D  B8 2020                      mov     ax,2020h
0250  B9 0021                      mov     cx,Appx_String_Len
0253  BF 014B R                    mov     di,offset Hex_String
0256  F3/ AB                       rep stosw
                                                        ;initialize pointers
                                                        ;and convert data
0258  BB 014B R                    mov     bx,offset Hex_String
025B  BA 0010                      mov     dx,size ASCII_String
025E  BF 017D R                    mov     di,offset ASCII_String
0261  EB 33 90                     jmp     Do_Line
0264  B9 004F       Output_Data:   mov     cx,String_Size  ;write a line to Std Output
0267  BA 0140 R                    mov     dx,offset Data_String
026A  E8 0341 R                    call    Output
                                                        ;set pointer to sixteen's
                                                        ;place of byte counter
026D  BF 0147 R                    mov     di,offset Byte_Counter+4
                                                        ;number of places in byte
0270  B9 0005                      mov     cx,5            ;counter minus one
0273                Adjust_Counter:                     ;increment input offset counter
0273  FE 05                        inc     byte ptr [di]   ;incr current number place value
0275  8A 05                        mov     al,[di]
0277  3C 39                        cmp     al,'9'          ;is it 9 or less?
0279  76 13                        jbe     Check_Line_Cnt  ;yes, check no. of lines output
027B  3C 41                        cmp     al,'A'          ;no is it larger than A?
027D  77 05                        ja      Check_For_F     ;yes, jump
027F  C6 05 41                     mov     byte ptr [di],'A'
0282  EB 0A                        jmp     short Check_Line_Cnt
0284  3C 46         Check_For_F:   cmp     al,'F'          ;is it 'F' or less?
0286  76 06                        jbe     Check_Line_Cnt  ;yes, check number of lines output
0288  C6 05 30                     mov     byte ptr [di],'0' ;no, make the digit '0'
028B  4F                           dec     di              ;move ptr to next higher place
028C  E2 E5                        loop    Adjust_Counter  ;caryy into next higher digit
                                                        ;check number of lines output
028E  FE 0E 0193 R  Check_Line_Cnt: dec    Line_Count      ;has block been output?
0292  75 B9                        jne     Do_Block        ;no, output another line
0294  EB AC                        jmp     Output_Lines    ;yes, output blank lines
                                                        ;place ASCII equivalent of
                                                        ;byte in string
0296  8A 04         Do_Line:       mov     al,[si]         ;get byte from buffer
0298  8A E8                        mov     ch,al           ;keep copy
029A  3C 20                        cmp     al,' '          ;control code?
029C  72 04                        jb      Not_Printable   ;yes, jump
029E  3C 7E                        cmp     al,'~'          ;no, is char a tilde or less?
02A0  76 02                        jbe     Printable       ;yes, use it
02A2  B0 2E         Not_Printable: mov     al,'.'          ;substitute '.' if not printable
02A4  AA            Printable:     stosb                   ;store into ASCII section of output
                                                        ;place hex equivalent of byte
                                                        ;in output string
```

(Continued on next page)

Listing Five

```
02A5  B4 02                          mov   ah,2             ;number of nibbles in a byte
02A7  87 FB                          xchg  di,bx            ;get hex section offset
02A9  B1 04       Swap_Nibbles:      mov   cl,4             ;size of nibble in bits
02AB  D2 C5                          rol   ch,cl            ;exchange nibbles
02AD  8A C5                          mov   al,ch            ;save copy
02AF  24 0F                          and   al,0fh           ;mask off high 4 bits
02B1  04 30                          add   al,'0'           ;convert to ASCII char.
02B3  3C 39                          cmp   al,'9'           ;is it '0-9'?
02B5  76 02                          jbe   Place_Digit      ;yes, store result
02B7  04 07                          add   al,7             ;put it into range 'A-F'

02B9  AA          Place_Digit:       stosb                  ;store into hex section of output
02BA  FE CC                          dec   ah               ;any more nibbles?
02BC  75 EB                          jnz   Swap_Nibbles     ;yes, convert again

                                                            ;reposition pointers
02BE  87 FB                          xchg  di,bx            ;get ASCII section offset
02C0  43                             inc   bx               ;skip space in hex section
02C1  46                             inc   si               ;move input pointer

                                                            ;input data when buffer empty
02C2  4D                             dec   bp               ;buffer used up yet?
02C3  75 03                          jnz   Check_Item_Cnt   ;no,check item count
02C5  E8 0324 R                      call  Input

                                                            ;check number of items stored
02C8  4A          Check_Item_Cnt:    dec   dx               ;store more data in string?
```

```
02C9  75 CB                        jnz     Do_Line          ;yes, get another byte
02CB  EB 97                        jmp     Output_Data      ;no, send a line

                                                            ;output last data string
                                                            ;if necessary
02CD  0B ED       Finish_Output:   or      bp,bp            ;was there any input?
02CF  75 27                        jnz     Error_3          ;no, send error message
02D1  B9 004F                      mov     cx,String_Size   ;yes get size & addr. of string
02D4  BA 0140 R                    mov     dx,offset Data_String
02D7  E8 0341 R                    call    Output           ;send the line

                                                            ;exit to DOS with ERRORLEVEL set
02DA  2A C0                        sub     al,al            ;return code = 0 for success

02DC  B4 4C       Exit_to_DOS:     mov     ah,Exit          ;terminate with AL=return code
02DE  CD 21                        int     21h

02E0  B9 001F     Error_1:         mov     cx,Pre_Dos_Size  ;wrong DOS version
02E3  BA 0194 R                    mov     dx,offset Pre_Dos_Error
02E6  BD 0001                      mov     bp,1             ;save ERRORLEVEL value
02E9  EB 2E 90                     jmp     Output_Err_Msg   ;send error message

02EC  B9 001C     Error_2:         mov     cx,Input_Size    ;input device error
02EF  BA 01B3 R                    mov     dx,offset Input_Error
02F2  BD 0002                      mov     bp,2             ;save ERRORLEVEL value
02F5  EB 22 90                     jmp     Output_Err_Msg   ;send error message

02F8  B9 001D     Error_3:         mov     cx,Empty_Size    ;empty input stream
02FB  BA 01CF R                    mov     dx,offset Empty_Error
02FE  BD 0003                      mov     bp,3             ;save ERRORLEVEL value
0301  EB 16 90                     jmp     Output_Err_Msg   ;send error message

0304  B9 001D     Error_4:         mov     cx,Output_Size   ;output device error
0307  BA 01EC R                    mov     dx,offset Output_Error
```

(Continued on next page)

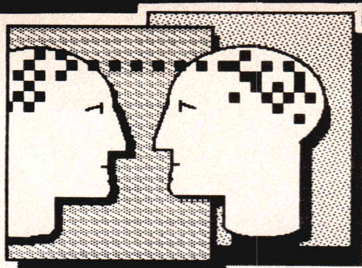
Listing Five

```
030A  BD 0004                   mov   bp,4              ;save ERRORLEVEL value
030D  EB 0A 90                  jmp   Output_Err_Msg    ;send error message

0310  B9 0016    Error_5:       mov   cx,Disk_Full_Size ;output device is full
0313  BA 0209 R                 mov   dx,offset Disk_Full_Error
0316  BD 0005                   mov   bp,5              ;save ERRORLEVEL value

0319             Output_Err_Msg:                        ;send message to Standard
                                                        ;Error device & pass return
                                                        ;code back to DOS
0319  B4 40                     mov   ah,Device_Output
031B  BB 0002                   mov   bx,Std_Err        ;use handle for Standard Error dev.
031E  CD 21                     int   21h
0320  8B C5                     mov   ax,bp             ;recover return code
0322  EB B8                     jmp   Exit_to_DOS       ;go terminate

0324             Input          proc  near              ;get data from Standard Input
0324  53                        push  bx                ;save hex section offset
0325  52                        push  dx                ;save item count
0326  B4 3F                     mov   ah,Device_Input
0328  BB 0000                   mov   bx,Std_Input      ;use handle for Standard Input
032B  B9 F000                   mov   cx,60*1024        ;size of input buffer
032E  BA 034F R                 mov   dx,offset Buffer;address of buffer for data
0331  CD 21                     int   21h
0333  72 B7                     jc    Error_2           ;jump if input device error
0335  0B C0                     or    ax,ax             ;any input?
```

```
0337  74 94                         jz      Finish_Output    ;no, send last string & exit
0339  BE 034F R                     mov     si,offset Buffer;set pointer to input
033C  8B E8                         mov     bp,ax            ;length of input
033E  5A                            pop     dx               ;restore item count
033F  5B                            pop     bx               ;restore hex section offset
0340  C3                            ret
0341                  Input         endp

0341                  Output        proc    near             ;send string to Standard Output
0341  B4 40                         mov     ah,Device_Output
0343  BB 0001                       mov     bx,Std_Output    ;use handle for Standard Output
0346  CD 21                         int     21h
0348  72 BA                         jc      Error_4          ;jump if output device error
034A  3B C1                         cmp     ax,cx            ;all requested bytes written?
034C  75 C2                         jne     Error_5          ;jump, output device full
034E  C3                            ret
034F                  Output        endp

= 034F                Buffer        equ     $                ;beginning of input buffer

034F                  Dump          endp

034F                  Code          ends


                                    end     Dump
```

End Listings

U.S. Robotics Modem Review



by Robert Blum

The CP/M Exchange RCP/M system is available for your use 24 hours a day, 7 days a week. Reach it by dialing (404) 449-6588.

Second only to the microcomputer itself, the modem is the most useful piece of equipment brought to us by the electronics revolution. Never before has information been so readily available, or so easily accessed as it is today. From your living room you can transfer funds among bank accounts, leave a purchase order to be executed by your broker the next day, or gather information from the database on your corporation's mainframe for that last minute report.

Modems aren't new. They have been a vital part of commercial data processing for years. Only recently, however, has the owner of a personal computer been able to afford one, generally, a manual 300-baud modem. Today all this has changed. There are now available numerous intelligent 1200-bps modems for the individual to choose from. As this review is being written, 2400-bps devices are being introduced that promise better transmission quality at twice the speed. Incredible as it may seem, there are already indications that the next generation of modems will feature speeds as fast as 9600 bps and cost little more than the current crop (see the article "Modems: 2400 Bit/Sec and Beyond" by Dale Walsh in the June '85 issue of *DDJ*).

Modems come in two basic forms: stand-alone and internal. Stand-alone modems are enclosed in a case that is separate from the computer. This case contains only the modem itself and the power supply needed for independent operation. Attachment to the computer is via a serial port and appropriate cabling.

If an extra serial port is not available or the system designer is forced to enclose the communications board in the same case as the rest of the system, then an internal modem is needed. An internal, or bus-oriented, modem does not require a serial port for connection to the computer. It plugs directly into a system's backplane and responds across the bus like any other system board. Thus, it provides both the serial interface and the modem functionality.

There is one distinct disadvantage to the bus-oriented modem: it cannot be moved between systems of different types. For example, if an S-100 machine is replaced by an IBM PC or PC compatible, the S-100 modem and any driving software is rendered useless. The choice between a standalone or internal modem depends heavily on the needs and preferences of the user.

The S-100 Modem

Producing a bus-oriented intelligent modem that maintains reliable 1200-bps operation has, to my knowledge, been accomplished only by U.S. Robotics. The U.S. Robotics S-100 modem is totally self-contained and plugs directly into any slot of the S-100 bus. Processor services are provided through two dip switch selectable I/O port addresses that furnish performance and software compatibility with the company's Password line of stand-alone modems.

The designer of the U.S. Robotics S-100 modem divided the two separate logic sections, serial port and intelligent modem, into two separate physical sections on the board. The serial interface, USART, and the necessary support circuits occupy the left side of the standard-size S-100 board. The right side is fitted with several 40-pin chips and a handful of other support devices that form the modem section.

The Modem

The modem circuit, on the right side of the board, is the functional equivalent of the U.S. Robotics Autodial 212A or Password modems. It is a fully functional auto-dial/auto-answer modem that operates at 0 to 300 baud and 1200 bps according to the 212A protocol. It consists of two custom programmed 8049 8-bit microprocessors. The first of these interacts with a 2921 signal-processing microprocessor element that is responsible for directly controlling the phone line and the conversion of the analog transmission data into digital form. The other handles the digital interface with the serial section as well as the execution and interpretation of controlling commands sent to, or status signals returned from, the modem section.

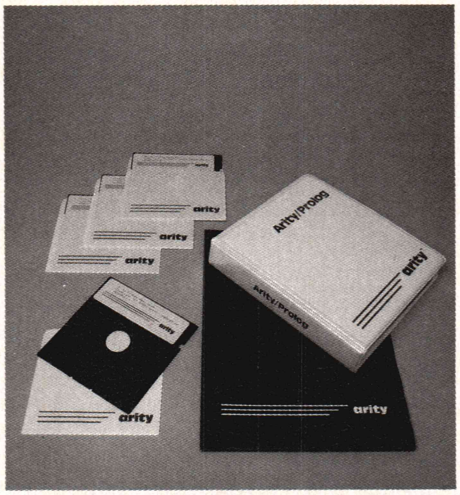
Commands are sent to the modem via the on-board serial interface in the form of upper case ASCII text characters, numbers, and special symbols. The modem control commands are a subset of the DC Hayes command set, the de facto standard.

Because the S-100 modem is command driven, an escape sequence of characters is provided to interrupt the modem during data transmission. On reset or power-up the default escape sequence is set to +++. To ensure that the escape sequence is found among other data being transmitted, it must be bracketed by a period of at least one second during which no data is transmitted. The escape sequence is more important than might at first appear: no other method of returning the modem to command mode is available outside

More Power Than You Thought Possible

Arity offers the first serious implementation of Prolog for IBM personal computers. Arity/Prolog is a powerful, highly optimized, and extended version of the logic programming language Prolog. Imagine building software applications with a language that solves problems through deduction and logical inference. The task of creating complex programs is much faster and easier, resulting in lower development costs. Arity/Prolog is now in use in a wide range of applications in industry, business, research, and education. The solution—the *Arity/Prolog Interpreter:*

- Source level debugger
- Virtual databases, each with a workspace of 16 megabytes
- Floating-point arithmetic
- String support for efficient text handling



- Interface to assembly language and 'C'
- Text screen manipulation
- Integrated programming shell to MSDOS
- Comprehensive set of evaluable predicates
- Definite clause grammar support

Arity/Prolog Interpreter $495.00

Arity also offers the *Arity/Prolog Compiler and Interpreter,* a sophisticated development environment for building AI applications. Essential for producing fast, serious production code.

Arity/Prolog Compiler and Interpreter $1950.00

The *Arity/Prolog Demo Disk* is available for $19.95. ■ Arity/Prolog products run on the IBM PC, XT, AT, and all IBM compatibles. ■ To order, call (617) 371-2422 or use the order form below.

arity corporation 358 Baker Avenue, Concord, MA 01742

Name

Organization

Address

Quantity	Product	Unit Price	Total Price
	Arity/Prolog Compiler & Interpreter	$1950.00	
	Arity/Prolog Interpreter	$ 495.00	
	Arity/Prolog Demo Disk	$ 19.95	
		Subtotal	
		MA residents add 5% sales tax	
		Total Amount	

☐ Enclosed is a check or money order to Arity Corporation

☐ Please bill my ☐ Mastercard ☐ Visa ☐ American Express

Account #

Valid ____/____ to ____/____ ______________ signature

☐ Please send me more information about Arity and Arity/Prolog

arity 358 Baker Avenue, Concord, MA 01742

M-AD-05

Circle **no. 16** on reader service card.

A Answer mode. Puts the modem into answer mode waiting for a ring.

A/ Reexecute the last command.

AT Attention. This two character sequence must proceed all commands.

D Dial the number that follows
- **P** Dial in pulse mode.
- **T** Dial in tone mode.
- **,** Delay two seconds

E Local echo. Echo all commands.
- **E0** Local echo off.
- **E1** Local echo on.

F Type of transmission.
- **F0** Half duplex.
- **F1** Full duplex.

M Loudspeaker control
- **M0** Loudspeaker off.
- **M1** Loudspeaker on until carrier lock.
- **M2** Loudspeaker always on.

Q Result code echo.
- **Q0** Result messages sent.
- **Q1** Result messages suppressed.

S Set controlling register.
- **S0=n** Answer on nth ring.
- **S2=n** Set the escape code to n.
- **S7=n** Length of time to wait for carrier.

V Verbal or Numeric result mode.
- **V0** Numeric mode.
- **V1** Verbal mode.

X Result code set.
- **X0** Standard result codes sent.
- **X1** Extended result codes sent.

Z Reset

Table 1
S-100 Modem Command Set

Result Codes

Verbal Mode	Numeric Mode
OK	0
CONNECT	1
RING	2
NO CARRIER	3
ERROR	4
CONNECT 1200	5

Table 2
S-100 Modem Result Codes

of dropping the DTR line.

When the escape sequence is seen the S-100 modem immediately disconnects from the phone line and returns to command mode. This action may not be desirable when you are communicating with systems that do not allow a return to local command mode while remaining on-line.

A complete list of all the commands recognized by the S-100 modem is contained in Table 1 (at left). The basic command form consists of the attention signal (AT) followed by one or more options and a single carriage-return character. For example, dialing a number and waiting for a carrier is accomplished with one command:

ATDT5551212

Here the leading AT is the attention sequence required of all commands. The D is the dial command followed by a T, which designates the tone dialing method. The rest of the command is the number to be dialed. The modem will dial the desired number without any further attention and will return the CONNECT or CONNECT 1200 result code if a connection is made. Otherwise, the NO CARRIER result code will be sent to indicate that no suitable connection was made in 17 seconds.

Acknowledgement of commands and the operating status of the modem are signaled by either full text messages or single digits sent following the execution of commands. Result codes are also sent to signal the controlling software of conditions such as the phone ringing and the presence or absence of a carrier. A complete list of the result codes is contained in Table 2 (at left).

The S-100 modem will also automatically switch between originate and answer modes. When answering the telephone, it will automatically choose either high- or low-speed communications. Result codes are then sent to the controlling software or terminal device to indicate whether a high- or low-speed connection has been established. The microcomputer can then adjust its own rate accordingly.

The S-100 modem comes with a speaker that reproduces the sound of the dialing tones and provides information about progress towards a connection. Once communication has been established, the speaker is automatically turned off if the appropriate option has been selected. Although the speaker is contained in the case of the computer, it is generally loud enough to be heard.

The Serial Interface

The serial interface, on the left side of the board, substitutes for the usual RS-232 connection between the computer and the modem. The 8251A USART provides the parallel S-100 bus interface and subsequent serialization of data and conversion of the baud rate to match that of the modem section. Communication with the USART thus takes place through two of the commonly used I/O channels of the 8080, Z80, and 8086 family of microprocessors.

The 8251A USART is a programmable serial communications device. Programming the 8251A requires the user to send commands to the 8251A. The 8251A is able to send information regarding the status of the communications to the microprocessor.

Installing the Modem

The first step in using the S-100 modem is to install it in the S-100 system. Because the S-100 modem uses the I/O port system of the microprocessor, the user must decide which ports to dedicate to the modem. One of the ports, the one with the lowest number, is used to transfer data to and from the serial interface. The other port is used to address the Mode/Command and Status registers of the 8251A. The S-100 modem is designed to use the two addresses at the top or bottom of any I/O port page. Thus, the modem may be addressed as 00,01; 0E,0F; 10,11; EE,EF and so on. This provides 32 different locations at which the modem's serial interface may be addressed.

Programming the S-100 Card Modem

Writing programs for the S-100 mo-

dem requires some knowledge of the 8251A USART and the command structures of that chip. Reading and writing data are fairly simple tasks for the 8251A. They merely require the use of the IN and OUT commands of the assembler along with the appropriate addresses. Setting up the USART for communications requires more skill.

The Mode/Command information is used to program the USART for the correct baud or bps rate and word length. Status information indicates whether the USART is ready to send or receive information. Errors are also indicated in the status register.

Listing One (at right) is a program fashioned after overlays used in the popular public-domain BYE or MBYE programs. It is a comprehensive example of how to implement all of the S-100 modem's features. A much simpler example could be written by following these steps:

1. Program the 8251A for the proper communications mode
2. Check the status for incoming characters
3. Check the status for errors
4. Get an incoming character
5. Process incoming character
6. Check the status for send ready
7. Send a character if status ready

Once the proper communications settings have been made, the program using the USART may send or receive information in any order desired. The status should always be checked before sending or receiving information.

Documentation

The manual accompanying the S-100 modem consists of 37 pages printed on one side. The first 20 pages provide information relevant only to the operation of the S-100 modem; the remainder of the manual is a reproduction of the 8251A data sheets. The documentation contains only enough information to get you started. The examples don't provide enough guidance on how to interact with the modem through commands or how to handle result codes. A large portion of the time I took writing the example found in Listing One was spent in front of the CRT as I used my debugger to solve the mysteries surrounding the workings of the command system.

User Impressions

I own two modems, the newest of which is the S-100 modem. During the past several months I have had the unique opportunity to switch between them when I met a communications problem. Never was I able to pinpoint a situation in which one performed better than the other, although I did find a few peculiarities in the S-100 modem that were bothersome. One particularly aggravating problem was that the S-100 had a nasty habit of answering the phone when no call had actually come in. The only other problem serious enough to deserve mention here is that, when dialing out at 300 baud, the originating carrier would be brought up immediately without waiting for the answer tone.

The Program Listing

I am placing this program in the public domain. If you find a problem or add an exciting feature, let me know so I can update the source code. You can download the program from my RCP/M (404-449-6588) or obtain a copy of it in one of a variety of formats (write me care of *DDJ*).

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 195**.

CP/M Exchange Listing (Text begins on page 102)

```
;************************************************************
;************************************************************
;
;       This routine is in the form of an overlay for the
;       popular MBYE - BYE public domain RCP/M programs.
;       Although not entirely compatible with those programs
;       only minor modifications would be necessary to adapt it
;       for use with them.  It is written in 8080 mnemonic to
;       insure maximum compatibility with multiple standard
;       assemblers and CPUs.
;
;************************************************************
;************************************************************

;************************************************************
;************************************************************
;
;       Define the equates for this routine
;
;************************************************************
;************************************************************

false:          equ     0
true:           equ     not false

md_res_ok:      equ     '0'                     ;result - OK
md_res_con300:  equ     '1'                     ;result - CONNECT (300)
md_res_ring:    equ     '2'                     ;result - RING
md_res_nocar:   equ     '3'                     ;result - NO CARRIER
md_res_con1200: equ     '5'                     ;result - CONNECT (1200)

md_mhz:         equ     60                      ;set to processor speed X 10

md_cr:          equ     0dh                     ;carriage return
md_lf:          equ     0ah                     ;line feed
md_eom:         equ     080h                    ;end of message

md_base_port:   equ     00h                     ;modem ports start at zero
md_data_port:   equ     md_base_port            ;data port address
md_stat_port:   equ     md_data_port+1          ;status port address

md_tr_rdy:      equ     01h                     ;transmitter empty
md_rcv_rdy:     equ     02h                     ;receiver ready
md_par_err:     equ     08h                     ;parity error
md_ovf_err:     equ     010h                    ;overflow error
md_frm_err:     equ     020h                    ;framing error
md_error:       equ     md_par_err+md_ovf_err+md_frm_err
                                                ;general mask for all errors
md_carrier:     equ     080h                    ;data carrier

md_mode_syn:    equ     0                       ;mode byte for synchronous
md_mode_x1:     equ     1h                      ;mode for async. X 1 clock
```

(Continued on next page)

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
md_mode_x16:    equ     2h                      ;mode for async. X 16 clock
md_mode_x64:    equ     3h                      ;mode for async. X 64 clock
md_mode_cl5:    equ     0h                      ;mode for character lnth of 5
md_mode_cl6:    equ     4h                      ;mode for character lnth of 6
md_mode_cl7:    equ     8h                      ;mode for character lnth of 7
md_mode_cl8:    equ     0ch                     ;mode for character lnth of 8
md_mode_nop:    equ     0h                      ;mode for no parity
md_mode_pon:    equ     010h                    ;mode for parity
md_mode_oddp:   equ     0h                      ;mode for odd parity
md_mode_evnp:   equ     020h                    ;mode for even parity
md_mode_stp1:   equ     040h                    ;mode for 1 stop bit
md_mode_stp15:  equ     080h                    ;mode for 1.5 stop bit
md_mode_stp2:   equ     0c0h                    ;mode for 2 stop bit

md_cmd_xmton:   equ     1h                      ;command for transmitter enable
md_cmd_xmtof:   equ     0h                      ;command for transmitter disable
md_cmd_dtron:   equ     2h                      ;command for DTR active
md_cmd_dtrof:   equ     0h                      ;command for DTR inactive
md_cmd_rcvon:   equ     4h                      ;command for receiver enable
md_cmd_rcvof:   equ     0h                      ;command for receiver disable
md_cmd_brkon:   equ     8h                      ;command for sending break
md_cmd_brkof:   equ     0h                      ;command for no break
md_cmd_eres:    equ     010h                    ;command for error reset
md_cmd_rtson:   equ     020h                    ;command for RTS active
md_cmd_rtsof:   equ     0h                      ;command for RTS inactive
md_cmd_ires:    equ     040h                    ;command for internal reset
md_cmd_hunt:    equ     080h                    ;command for hunt mode (not used)

outstr:         macro   string
                if      not nul string
                lxi     h,string                ;point at string to output
                endif
                call    md_out_msg              ;call subroutine
                mend

outchr:         macro   byte
                if      not nul byte
                mvi     c,byte                  ;load C with byte value
                endif
                call    md_out                  ;output one byte
                mend

tstngo:         macro   test,good,error
                mov     c,a                     ;save in C
                cpi     test                    ;is it an test
                if not nul good
                jz      good                    ;yes - take good exit
                endif
                if not nul error
                jnz     error                   ;no - take error exit
                mend

sloout:         macro   port
                if      md_mhz gt 40
                rept    ((md_mhz-40)/10)-1

                xchg                            ;allow 8251-A to catch up
                xchg                            ;*
                mend
                endif
                out     port
                mend

sloin:          macro   port
                if      md_mhz gt 40
                rept    ((md_mhz-40)/10)-1
                xchg                            ;allow 8251-A to catch up
                xchg                            ;*
                mend
                endif
                in      port
                mend

                page
;***************************************************************
;***************************************************************
;
;       This routine initializes the modem to the defaults
;       set after a hardware reset.
;
;***************************************************************
;***************************************************************

md_init:

                call    md_carck                ;is carrier still up
                cnz     md_hangup               ;yes - drop the line
                call    md_set1200              ;default to 1200 baud
                call    md_reset_modem          ;reset the MODEM

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
```

(Continued on page 108)

BRIEF™

"BRIEF, The Programmer's Editor, is simply the best text editor you can buy."

John Dvorak, InfoWorld 7/8/85

Choosing the right program editor is an extremely important decision. After all, more time is spent with your program editor than with any other category of software. A good program editor helps to stimulate creativity and produce programs of superior design. A poor program editor, on the other hand, can be worse than none at all. It can slow you down, get in your way, and make even the simplest tasks difficult.

BRIEF - PROGRAM EDITING *YOUR WAY*

Every programmer has an individual style that makes their work unique. Most program editors compromise that style by forcing the programmer to conform to their methods. Not so with BRIEF.

BRIEF's most powerful feature is its ability to conform to your way of programming. BRIEF can be easily tailored/customized to your individual preferences. For example, keyboard reassignment allows the keyboard to be used in whatever way you prefer. Keystroke macros allow long sequences of editor directives to be repeated by pressing a single key. Plus you can simultaneously work with numerous program and data files, and configure many windows to control BRIEF's visual presentation.

75% OF THE EXPERTS WHO HAVE TRIED BRIEF SWITCHED!

(Call For Details)

"A BONA FIDE UNDO" Steve McMahon – BYTE REVIEW "BUILD YOUR DREAM EDITOR" MARCH 1985

Here are Steve McMahon's exact words: "...BRIEF implements a true undo facility, by default allowing command-by-command recovery from the last 30 commands...The number of commands you want to be able to undo can be changed." (up to 300) "Only with BRIEF, though, was it possible to undo a macro that produced 4000 words of text with a single keystroke."

EVERY FEATURE YOU WANT

BRIEF supports practically every feature you've ever seen, and some you probably haven't thought of yet.

- Edit Multiple Large Files
- Windows (Tiled & Pop-up)
- Unlimited File Size
- Full UNDO (N Times)
- True Automatic Indent for C
- Exit to DOS Inside BRIEF
- Compile programs inside BRIEF
- Uses All Available Memory
- Intuitive Commands
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning
- Reconfigurable Keyboard
- Online Help
- Search for "Regular Expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- And ... a Complete, Compiled, Programmable and Readable Macro Language.

```
==================================== make.c ====================================
|int      handle = 0;
|
|main (argc, argv█
|
|int            argc;
=================== fsa.h ====================== makefile.h =====================
|#include "..\include\ctype.h"        |/**
|                                     | **                makefile.h:
|typedef  struct                      | **
|{                                    | **          This is the definitions fil
|   short    action,                  | **   Hopefully, it won't be unreasonab
|            state;                   | **   that have been written.
|} Fsa;                               | **/
|#ifdef    FSA_MAIN                   |
|Fsa       fsa[][8] = {/* Alphanum  Co|typedef struct cmd_struct
|                                     |{
|/* State 0. */           0, 2,     10|   char        *cmd_text;
|/* State 1. */           10, 0,    10|
|/* State 2. */           0, 2,     1,|   struct cmd_struct *next_cmd;
|/* State 3. */           0, 5,     11|} *Cmd_Ptr, Cmd;
|/* State 4. */           0, 4,     0,|
Mismatched open parenthesis.              Line: 11   Col: 17        2:17 pm
```

A TYPICAL BRIEF SCREEN

Notice there are three windows on the screen simultaneously and each one is viewing a different file. The mainline of a C program is visible in the uppermost window; the programmer has run a syntax checking macro which found a mismatched open parenthesis in the arguments to the mainline. The other two windows show header files containing information crucial to the design of the program. BRIEF can have an unlimited number of windows and files accessed simultaneously.

BRIEF'S ONLY LIMITATIONS ARE THE ONES *YOU* SET

Full refund if not satisfied in 30 days. ONLY $195

AVAILABILITY: IBM-PC and Compatibles, AT, & TANDY 2000

→ **FREE** with order: "Best of BRIEF Macros" — Contest entries include macros for Fortran, C, Calculator, Base Converter, etc. Call before November 30. **800-821-2492**

Solution Systems™

335-D Washington St., Norwell, MA 02061 (617) 659-1571 **800-821-2492.**

Steve McMahon's quote courtesy Suntype Publishing Systems. BYTE review by Mr. McMahon may be found in BYTE Magazine March 1985.

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
;       This subroutine exits with the phone on hook.
;
;*************************************************************
;*************************************************************

md_hangup:

                call    md_wait_1               ;wait one second
                lxi     h,md_dis_msg            ;point at disconnect message
                call    md_out_msg              ;write string to modem
                call    md_wait_1               ;wait one second
                call    md_carck                ;carrier should have dropped
                rz                              ;it did - return
                jp      md_hangup               ;error recovery may be improved
                                                ;for now just continue telling
                                                ;the modem to hangup

;*************************************************************
;*************************************************************
;
;       Dial a number
;
;*************************************************************
;*************************************************************

md_dial:

                outstr  md_attn_ldin            ;send AT
                outstr  md_dial_ldin            ;send D
                outstr  md_tone_stub            ;send T
                outstr  md_number               ;send number
                call    md_read_res             ;get the result code

                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       The carrier check routine will return the Z condition
;       code if the carrier is not present.
;
;*************************************************************
;*************************************************************

md_carck:

                call    md_status               ;get modem status
                ani     md_carrier              ;strip all but carrier bit

                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       Output a control string to the modem.  The string must
;       end with a 080h.
;
;*************************************************************
;*************************************************************

md_out_msg:

                mov     c,m                     ;get a byte
                mov     a,c                     ;is this end of string?
                rlc                             ;*
                rc                              ;yes - return
                call    md_out                  ;output it
                inx     h                       ;point at next byte
                jmp     md_out_msg              ;loop until complete

;*************************************************************

;*************************************************************
;
;       Output one byte to the modem
;
;*************************************************************
;*************************************************************

md_out:

                call    md_status               ;get modem status
                ani     md_tr_rdy               ;is transmitter ready to
                                                ;receive a byte
                jz      md_out                  ;no - loop until ready
                mov     a,c                     ;put data byte into A
                out     md_data_port            ;output it
                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       Input one byte from the modem
;
;*************************************************************
;*************************************************************

md_input:
```

```
        call    md_status               ;get USART status
        ani     md_rcv_rdy              ;is a byte available
        jz      md_input                ;no - loop until ready
        in      md_data_port            ;get data byte
        ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Input a verbal or numeric response message from the
;       modem and convert to a numeric response in the
;       A register.
;
;*****************************************************************
;*****************************************************************

md_read_res:

        call    md_input                ;get the leading CR
        tstngo  md_cr,md_read_res_ver   ;if CR go to verbal routine

        sta     save_res_code   ;save result code for exit
        call    md_input                ;get the trailing CR
        tstngo  md_cr,md_read_res_ext2,md_read_res_err
                                        ;go to exit if CR otherwise
                                        ;go to error routine
md_read_res_ver:

        call    md_input                ;get the leading LF
        tstngo  md_lf,,md_read_res_err  ;if not LF go to error routine

        call    md_input                ;get the first response byte

        tstngo  'O',md_read_res_ok      ;is it OK
        tstngo  'C',md_read_res_con     ;is it CONNECT or CONNECT 1200
        tstngo  'R',md_read_res_ring    ;is it RING
        tstngo  'N',md_read_res_ncar    ;is it NO CARRIER

;
;       Some type of unexpected error has occurred.  An unknown result
;       message has been sent.  Simply empty the USART and return with
;       ERROR posted in the C register.
;
md_read_res_err:

        call    md_wait_1               ;delay one second
        call    md_status               ;has any further data been sent
        ani     md_rcv_rdy              ;*
        jnz     md_read_res_err         ;yes - loop until empty

        mvi     a,'4'                   ;set error code in A
        sta     save_res_code           ;save for exit
        jmp     md_read_res_ext2        ;return to caller
md_read_res_ok:

        call    md_input                ;get the K
        tstngo  'K',,md_read_res_err    ;is it really a K
        mvi     a,'0'                   ;set our numeric result code
        sta     save_res_code           ;*
md_read_res_ext:

        call    md_input                ;get the trailing CR
        tstngo  md_cr,,md_read_res_err  ;is it really a CR
md_read_res_ext1:

        call    md_input                ;get the trailing LF
        tstngo  md_lf,,md_read_res_err  ;is it really a LF
md_read_res_ext2:

        mvi     a,1                     ;wait time of .1 second
        call    md_wait_1a              ;allow modem to recover
        lda     save_res_code           ;put user result code into A
        ret                             ;return to caller
md_read_res_con:

        call    md_input                ;get the O
        tstngo  'O',,md_read_res_err    ;is it really a O
        call    md_input                ;get the N
        tstngo  'N',,md_read_res_err    ;is it really a N
        call    md_input                ;get the N
        tstngo  'N',,md_read_res_err    ;is it really a N
        call    md_input                ;get the E
        tstngo  'E',,md_read_res_err    ;is it really a E
        call    md_input                ;get the C
        tstngo  'C',,md_read_res_err    ;is it really a C
        call    md_input                ;get the T
        tstngo  'T',,md_read_res_err    ;is it really a T

        mvi     a,'1'                   ;set the 300 baud result
        sta     save_res_code           ;*

        call    md_input                ;get the trailing CR or blank
        tstngo  ' ',,md_read_res_ext1   ;is it really a BLANK
        call    md_input                ;get the 1
        tstngo  '1',,md_read_res_err    ;is it really a 1
        call    md_input                ;get the 2
        tstngo  '2',,md_read_res_err    ;is it really a 2
```

(Continued on next page)

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
                call    md_input                ;get the 0
                tstngo  '0',,md_read_res_err    ;is it really a 0
                call    md_input                ;get the 0
                tstngo  '0',,md_read_res_err    ;is it really a 0

                mvi     a,'5'                   ;set the 1200 baud result code
                sta     save_res_code           ;*
                jmp     md_read_res_ext         ;exit thru common routine

md_read_res_ring:

                call    md_input                ;get the I
                tstngo  'I',,md_read_res_err    ;is it really a I
                call    md_input                ;get the N
                tstngo  'N',,md_read_res_err    ;is it really a N
                call    md_input                ;get the G
                tstngo  'G',,md_read_res_err    ;is it really a G

                mvi     a,'2'                   ;set ring result code
                sta     save_res_code           ;*
                jmp     md_read_res_ext

md_read_res_ncar:

                call    md_input                ;get the O
                tstngo  'O',,md_read_res_err    ;is it really a O
                call    md_input                ;get the blank
                tstngo  ' ',,md_read_res_err    ;is it really a BLANK
                call    md_input                ;get the C
                tstngo  'C',,md_read_res_err    ;is it really a C
                call    md_input                ;get the A
                tstngo  'A',,md_read_res_err    ;is it really a A
                call    md_input                ;get the R
                tstngo  'R',,md_read_res_err    ;is it really a R
                call    md_input                ;get the R
                tstngo  'R',,md_read_res_err    ;is it really a R
                call    md_input                ;get the I
                tstngo  'I',,md_read_res_err    ;is it really a I
                call    md_input                ;get the E
                tstngo  'E',,md_read_res_err    ;is it really a E
                call    md_input                ;get the R
                tstngo  'R',,md_read_res_err    ;is it really a R

                mvi     a,'3'                   ;set no carrier result code
                sta     save_res_code           ;*
                jmp     md_read_res_ext         ;return to user

;****************************************************************
;****************************************************************
;
;       Get USART status and return in A
;
;****************************************************************
;****************************************************************

md_status:

                in      md_stat_port            ;get modem status
                ret                             ;return to caller

;****************************************************************
;****************************************************************
;
;       This routine is a duplication of one that already exists
;       in the MBYE or BYE programs.  It is included here to make
;       this routine complete within itself.
;
;****************************************************************
;****************************************************************

md_wait_1:

                mvi     a,10                    ;constant to delay for 1 second

md_wait_la:

                push    psw                     ;save loop counter
                call    md_delay                ;delay .1 second
                pop     psw                     ;restore loop counter
                dcr     a                       ;is loop complete
                jnz     md_wait_la              ;no - loop until complete

                ret                             ;yes - return to caller

;****************************************************************
;****************************************************************
;
;       This subroutine will delay returning for .1 of a second
;       depending on the processor speed.
;
;****************************************************************
;****************************************************************

md_delay:
```

(Continued on page 112)

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
                lxi     b,(4167*(md_mhz/10)) + (417*(md_mhz mod 10))
                                                ;constant times mod 10 MHZ
md_delay_a:
                dcx     b                       ;decrement loop counter
                mov     a,b                     ;has counter elapsed
                ora     c                       ;*
                jnz     md_delay_a              ;*

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
;       Ask the MODEM to respond
;
;***************************************************************
;***************************************************************

md_wake_up:

                outstr  md_attn_ldin            ;send AT
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get result code

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
;       Reset the MODEM to it's default values
;
;***************************************************************
;***************************************************************

md_reset_modem:

                outstr  md_attn_ldin            ;send AT
                outstr  md_reset_stub           ;send Z
                outstr  md_eom_stub             ;send CR
                call    md_wait_1               ;*
                mvi     a,'0'                   ;set a result code of OK

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
;       Set the MODEM to not answer the phone if it rings
;
;***************************************************************
;***************************************************************

md_set_noans:

                outstr  md_attn_ldin            ;send AT
                outstr  md_setreg_ldin          ;send S
                outstr  md_anscnt_stub          ;send 0=
                outchr  '0'                     ;send no rings
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
;       Set the MODEM to answer the phone after 1 ring
;
;***************************************************************
;***************************************************************

md_set_ans1:

                outstr  md_attn_ldin            ;send AT
                outstr  md_setreg_ldin          ;send S
                outstr  md_anscnt_stub          ;send 0
                outchr  '1'                     ;send one ring
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;***************************************************************
;***************************************************************
;
;       Set the MODEM to ECHO all control characters
;
;***************************************************************
;***************************************************************

md_set_echo_on:

                outstr  md_attn_ldin            ;send AT
                outstr  md_echo_ldin            ;send E
```

```
                outstr  md_eon_stub             ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM to not ECHO any control characters
;
;*****************************************************************
;*****************************************************************

md_set_echo_off:

                outstr  md_attn_ldin            ;send AT
                outstr  md_echo_ldin            ;send E
                outstr  md_eoff_stub            ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM to FULL duplex operation
;
;*****************************************************************
;*****************************************************************

md_set_full_on:

                outstr  md_attn_ldin            ;send AT
                outstr  md_duplx_ldin           ;send F
                outstr  md_fdup_stub            ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM to HALF duplex operation
;
;*****************************************************************
;*****************************************************************

md_set_half_on:

                outstr  md_attn_ldin            ;send AT
                outstr  md_duplx_ldin           ;send F
                outstr  md_hdup_stub            ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM speaker on
;
;*****************************************************************
;*****************************************************************

md_set_spkr_on:

                outstr  md_attn_ldin            ;send AT
                outstr  md_spkr_ldin            ;send F
                outstr  md_spkon_stub           ;send 2
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM speaker off
;
;*****************************************************************
;*****************************************************************

md_set_spkr_off:

                outstr  md_attn_ldin            ;send AT
                outstr  md_spkr_ldin            ;send F
                outstr  md_spkoff_stub          ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM speaker on during dialing
```

(Continued on next page)

CP/M Exchange Listing

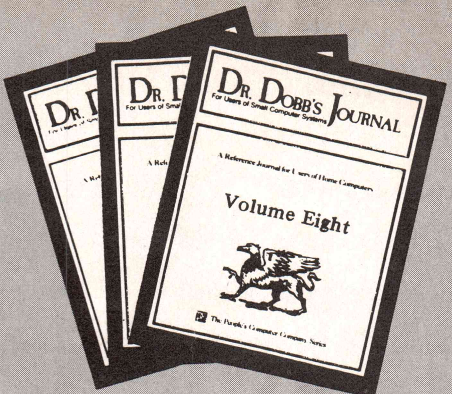
(Listing continued, text begins on page 102)

```
;
;*****************************************************************
;*****************************************************************

md_set_spkr_dial:

                outstr  md_attn_ldin            ;send AT
                outstr  md_spkr_ldin            ;send F
                outstr  md_spkdl_stub           ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM result messages on
;
;*****************************************************************
;*****************************************************************

md_set_res_on:

                outstr  md_attn_ldin            ;send AT
                outstr  md_reslt_ldin           ;send Q
                outstr  md_reson_stub           ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM result messages off
;
;*****************************************************************
;*****************************************************************

md_set_res_off:

                outstr  md_attn_ldin            ;send AT
                outstr  md_reslt_ldin           ;send Q
                outstr  md_resoff_stub          ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM result messages to numeric
;
;*****************************************************************
;*****************************************************************

md_set_res_num:

                outstr  md_attn_ldin            ;send AT
                outstr  md_resmod_ldin          ;send V
                outstr  md_num_stub             ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM result messages to verbal
;
;*****************************************************************
;*****************************************************************

md_set_res_ver:

                outstr  md_attn_ldin            ;send AT
                outstr  md_resmod_ldin          ;send V
                outstr  md_ver_stub             ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*****************************************************************
;*****************************************************************
;
;       Set the MODEM result messages to extended format
;
;*****************************************************************
;*****************************************************************
```

(Continued on page 116)

CONTINUING THE TRADITION

DR. DOBB'S JOURNAL ANNOUNCES THE RELEASE OF BOUND VOLUME 8



Every 1983 issue together in one source

BOUND VOLUME 8 (Item #020)
DDJ turns pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. Jim Hendrix's Small C compiler. Ed Ream's RED screen editor. A microcomputer subset of the Defense Department's official programming language, Ada. C and Forth and 68000 software. Because the magazine increased in size in 1983, this volume is bigger and better that ever.

Vol. 1 1976 (Item #013)
The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming which was then the only way to do things. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing.
Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI, and more.

Vol. 2 1977 (Item #014)
1977 found DDJ still on the forefront. These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 6080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NIBL.
Articles are about Lawrence Livermoore Lab's BASIC, Alpha Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities, and even more.

Vol. 3 1978 (Item #015)
The microcomputer industry entered into its adolescence in 1978. This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today.
Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors, and much, much more.

Vol. 4 1979 (Item #016)
This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages, innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080IZ80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference.
Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Psuedo-random Sequences, and Interfacing a Micro to a Mainframe—more than ever!

Vol. 5 1980 (Item #017)
All the ground-breaking issues from 1980 in one volume! Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full.
Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080. Further with Tiny BASIC, a Syntax-Oriented Compiler Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler and, as always, even more!

Vol. 6 1981 (Item #018)
1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features.
Highlights: Information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982 (Item #019)
In 1982 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler, and we continue to publish utility programs and useful algorithms. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi supplied a year of "Clinic" columns while delivering his famous review of JRT Pascal and writing the first serious technical comparison of CP/M-86 and MSDOS. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond, in the provocative essay, "Fifth-generation Computers."

Complete your reference library. Buy the entire set of Dr. Dobb's Journals from 1976 through 1983, Bound Volumes 1–8, for $195.00. That's $34.00 off the combined individual prices—a savings of almost 15%!

To order by credit card, call toll free: 1-800-528-6050 ext. 4001. Refer to the item number of each volume. Or mail this coupon to: *Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303.*

YES! ☐ Please send me the following Volumes of **Dr. Dobb's Journal.**
Payment must accompany your order.
Please charge my: ☐ Visa ☐ MasterCard ☐ American Express
I enclose ☐ Check/money order

Card # ____________ Expiration Date ____________
Signature ____________
Name ____________ Address ____________
(please, no P.O. Boxes)
City ____________ State ____ Zip ____________

Mail to Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303

Vol. 1	______	x $27.75 =	______
Vol. 2	______	x $27.75 =	______
Vol. 3	______	x $27.75 =	______
Vol. 4	______	x $27.75 =	______
Vol. 5	______	x $27.75 =	______
Vol. 6	______	x $27.75 =	______
Vol. 7	______	x $30.75 =	______
Vol. 8	______	x $31.75 =	______
All 8	______	x $195.00 =	______
		Sub-total $	______

California residents add applicable sales tax____% ______

Postage & Handling Must be Included with order.
Please add $1.75 per book in U.S. ($4.25 each surface mail outside U.S. Foreign Airmail rates available on request.)

TOTAL $ ______

3109B

Allow 6–9 weeks for delivery.

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
md_set_res_ext:

                outstr  md_attn_ldin            ;send AT
                outstr  md_res_ldin             ;send X
                outstr  md_extres_stub          ;send 1
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       Set the MODEM result messages to standard format
;
;*************************************************************
;*************************************************************

md_set_res_std:

                outstr  md_attn_ldin            ;send AT
                outstr  md_res_ldin             ;send X
                outstr  md_stdres_stub          ;send 0
                outstr  md_eom_stub             ;send CR
                call    md_read_res             ;get the result code in A

                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       Reset the USART.
;
;       This routine has been written for the worst case reset
;       condition.  Upon initial power on the 8251A may be in
;       an undetermined mode state.  For that reason the maximum
;       mode setup case of setting synchronous mode and two sync
;       characters is used.  Otherwise, there would be a
;       possibility of not consistently completing a reset.
;
;       When exit from this routine is taken the 8251A will be
;       ready for the initial mode and command - command bytes.
;
;*************************************************************
;*************************************************************

md_reset_usart:

                xra     a                       ;get a zero in A

                sloout  md_stat_port            ;output mode synchronous

                sloout  md_stat_port            ;output first sync byte

                sloout  md_stat_port            ;output second sync byte

                mvi     a,md_cmd_ires           ;8251A reset command
                sloout  md_stat_port            ;issue the reset command
                ret                             ;return to user

;*************************************************************
;*************************************************************
;
;       Reset the USART error status
;
;*************************************************************
;*************************************************************

md_reset_err:

                lda     save_cur_cmd            ;get error reset command
                sloout  md_stat_port            ;send to status port
                ret                             ;return to caller

;*************************************************************
;*************************************************************
;
;       Set the USART to 1200 baud
;
;*************************************************************
;*************************************************************

md_set1200:

                call    md_reset_usart          ;establish known initial mode

                mvi     a,md_mode_x16+md_mode_cl8+md_mode_nop+md_mode_stpl
                                                ;set mode to:
                                                ;               16X clock
                                                ;               8 bits per characte
                                                ;
                                                ;               no parity
                                                                1 stop bit
                sta     save_cur_mode           ;save current mode byte
                sloout  md_stat_port            ;send the mode byte
```

```
            mvi     a,md_cmd_xmton+md_cmd_dtron+md_cmd_rcvon+md_cmd_rtson+md_cm
                                                ;set command to:
                                                ;               transmitter on
                                                ;               DTR on (not used)
                                                ;               receiver on
                                                ;               RTS on
                                                ;               RESET errors
            sta     save_cur_cmd                ;save current command byte
            sloout  md_stat_port                ;send the command byte
            ret                                 ;return to caller

;*************************************************************
;*************************************************************
;
;       Set the USART to 600 baud
;
;*************************************************************
;*************************************************************

md_set600:

            call    md_reset_usart              ;establish known initial mode

            mvi     a,md_mode_x16+md_mode_cl8+md_mode_nop+md_mode_stpl
                                                ;set mode to:
                                                ;               16X clock
                                                ;               8 bits per characte
                                                ;               no parity
                                                ;               1 stop bit
            sta     save_cur_mode               ;save current mode byte
            sloout  md_stat_port                ;send the mode byte

            mvi     a,md_cmd_xmton+md_cmd_dtron+md_cmd_rcvon+md_cmd_rtsof+md_cm
                                                ;set command to:
                                                ;               transmitter on
                                                ;               DTR on (not used)
                                                ;               receiver on
                                                ;               RTS off
                                                ;               RESET errors
            sta     save_cur_cmd                ;save current command byte
            sloout  md_stat_port                ;send the command byte
            ret                                 ;return to caller

;*************************************************************
;*************************************************************
;
;       Set the USART to 300 baud
;
;*************************************************************
;*************************************************************


md_set300:

            call    md_reset_usart              ;establish known initial mode

            mvi     a,md_mode_x64+md_mode_cl8+md_mode_nop+md_mode_stpl
                                                ;set mode to:
                                                ;               16X clock
                                                ;               8 bits per characte
                                                ;               no parity
                                                ;               1 stop bit
            sta     save_cur_mode               ;save current mode byte
            sloout  md_stat_port                ;send the mode byte

            mvi     a,md_cmd_xmton+md_cmd_dtron+md_cmd_rcvon+md_cmd_rtson+md_cm
                                                ;set command to:
                                                ;               transmitter on
                                                ;               DTR on (not used)
                                                ;               receiver on
                                                ;               RTS on
                                                ;               RESET errors
            sta     save_cur_cmd                ;save current command byte
            sloout  md_stat_port                ;send the command byte
            ret                                 ;return to caller

;*************************************************************
;*************************************************************
;
;       Set the USART to 150 baud
;
;*************************************************************
;*************************************************************

md_set150:

            call    md_reset_usart              ;establish known initial mode
            mvi     a,md_mode_x64+md_mode_cl8+md_mode_nop+md_mode_stpl
                                                ;set mode to:
                                                ;               16X clock
                                                ;               8 bits per characte
                                                ;               no parity
                                                ;               1 stop bit
            sta     save_cur_mode               ;save current mode byte
            sloout  md_stat_port                ;send the mode byte

            mvi     a,md_cmd_xmton+md_cmd_dtron+md_cmd_rcvon+md_cmd_rtsof+md_cm
                                                ;set command to:
                                                ;               transmitter on
                                                ;               DTR on (not used)
                                                ;               receiver on
                                                ;               RTS off
                                                ;               RESET errors
            sta     save_cur_cmd                ;save current command byte
            sloout  md_stat_port                ;send the command byte
            ret                                 ;return to caller
```

(Continued on next page)

CP/M Exchange Listing

(Listing continued, text begins on page 102)

```
                page

;*******************************************************************
;*******************************************************************
;
;       This section of the program falls after all other
;       program code as required by the relocation routine used.
;
;*******************************************************************
;*******************************************************************

endobj:

md_dis_msg:      db      '+++',md_eom          ;disconnect from phone line
md_repeat_msg:   db      'A/',md_eom           ;repeat last command

md_attn_ldin:    db      'AT',md_eom           ;beginning of all messages

md_dial_ldin:    db      'D',md_eom            ;dial number lead-in
md_echo_ldin:    db      'E',md_eom            ;echo all commands lead-in
md_duplx_ldin:   db      'F',md_eom            ;duplex transmission
md_spkr_ldin:    db      'M',md_eom            ;speaker control
md_reslt_ldin:   db      'Q',md_eom            ;result message control
md_setreg_ldin:  db      'S',md_eom            ;set registers
md_resmod_ldin:  db      'V',md_eom            ;set numeric/verbal result mode
md_res_ldin:     db      'X',md_eom            ;set result mode

md_ans_stub:     db      'A',md_eom            ;answer phone immediately
md_pulse_stub:   db      'P',md_eom            ;dial via pulse method
md_tone_stub:    db      'T',md_eom            ;dial via tone method
md_pause_stub:   db      ',',md_eom            ;delay for 2 seconds during
dialing
md_eoff_stub:    db      '0',md_eom            ;turn local echo off
md_eon_stub:     db      '1',md_eom            ;turn local echo on
md_hdup_stub:    db      '0',md_eom            ;half duplex
md_fdup_stub:    db      '1',md_eom            ;full duplex
md_spkoff_stub:  db      '0',md_eom            ;turn speaker off
md_spkdl_stub:   db      '1',md_eom            ;turn speaker on until connect
md_spkon_stub:   db      '2',md_eom            ;turn speaker on
md_reson_stub:   db      '0',md_eom            ;send result messages
md_resoff_stub:  db      '1',md_eom            ;suppress result messages
md_anscnt_stub:  db      '0=',md_eom           ;set answer count
md_escod_stub:   db      '2',md_eom            ;set escape code
md_carwat_stub:  db      '7',md_eom            ;set wait for carrier time
md_num_stub:     db      '0',md_eom            ;result messages are numeric
md_ver_stub:     db      '1',md_eom            ;result messages are verbal
md_stdres_stub:  db      '0',md_eom            ;standard result codes
md_extres_stub:  db      '1',md_eom            ;extended result codes
md_reset_stub:   db      'Z',md_eom            ;reset command
md_eom_stub:     db      md_cr,md_eom          ;end of message

md_number:       ds      20                    ;place phone number here
                                               ;followed by md_eom_stub
save_res_code:   ds      1


save_cur_mode:   ds      1                     ;save area for current mode byte
save_cur_cmd:    ds      1                     ;save area for current command byte
```

Listing 1 - S-100 Modem Overlay Program

End Listing

WorkArea

A Text Management & Presentation System *for Software Developers*

- WorkArea incorporates your word-processor (or the full function editor provided) into an object-oriented data-base containing text, numbers, strings, vectors, calculations, scripts, windows, interrogatives and more.
- WorkArea can be used as an authoring system, for building minor expert or knowledge-base systems.
- WorkArea can be used in conjunction with DMCQ consulting services to build customized products. WorkArea is written in 'c' and among its utilities has functions for compressing files (directly readable by WorkArea) and merging WorkArea files.
- WorkArea is available in read-only versions.
- WorkArea runs on the IBM PC and compatibles with 256K RAM, DOS 2.xx, floppy or fixed-disk, monochrome or color monitor.

Complete WorkArea package: $275

Demo and documentation package: $30

(site license or special orders possible)

D M C Q

Systems Consulting and Software Development

To order: P.O. Box 602 (707) 743-1907
Ukiah, CA 95482

Send a check or money order. Sorry, we do NOT accept phone, credit card or COD orders. Please supply a street address for UPS delivery.

Circle **no. 21** on reader service card.

WORD TECH SYSTEMS

dBIII COMPILER

A Compiler For dBASE III
$600

Write-Hand Man
(Sidekick for CP/M)
8" Format $49.95 5¼" Formats $54.95

CP/M LIVES!

A die hard CP/M house actually exists. We confess, we still sell, support, and use CP/M systems and software. Custom multiuser hardware and dBASE II applications are available.

d/MULTI

Multiuser dBASE II for TurboDOS

At last! File & record locking as easy as d-BASE-II. Now unlimited users can perform the magic of dBASE in the program or interactive mode. Don't look for it anywhere else — A MARTIAN EXCLUSIVE!

d/MULTI . . . $695

Dealer inquiries encouraged

Martian Technologies
8348 Center Dr., Ste. F, La Mesa, CA 92041
(619) 464-2924

Circle **no. 74** on reader service card.

NEON™: TURN ON THE FULL POWER OF YOUR MACINTOSH.™

Hidden within your Mac is the programming power, flexibility, and speed to match your imagination. Neon is your key to this object-oriented world. Based on the same design philosophy as the Mac itself, Neon lets you create and command objects — program modules that you build, perfect, and add to your personal collection of tools. In this Smalltalk-like environment, you wield the power to quickly assemble and test ideas, tuning as you go for maximum speed and efficiency. □ Neon works in 128K or dynamically expands for the fattest Mac possible. It is easy to work with from the very beginning. Additional help is provided through the comprehensive manual by Danny Goodman. □ Created by Kriya Systems, Inc. for the development of our Typing Tutor III™ program, Neon is your answer for professional software development. With Neon, there are no licensing fees. Ever. For the MacIntosh developer, Neon is the best choice.

To order your copy of Neon, call 1-800-34KRIYA now with Visa or MasterCard, or send your check to Kriya Systems, Inc., Six Export Drive, Sterling, VA 22170. The price is $155, including shipping and handling.

Neon and Typing Tutor III are trademarks of Kriya Systems, Inc. Apple Macintosh is a trademark of Apple Computers, Inc.

Circle **no. 9** on reader service card.

ANNOUNCING! DR. DOBB'S COMPLETE TOOLBOX OF C

Dr. Dobb's Journal, the most respected source of technical software information available, brings you this collection of powerful programming tools.

Available in October from M&T Publishing and Brady Communications—

Dr. Dobb's Toolbook for C (Item #005)

A comprehensive library of valuable C code

Many of Dr. Dobb's most popular articles on C from sold-out issues are updated and reprinted in this unique reference, along with new C programming tools. **THE TOOLBOOK** contains a complete C compiler, an assembler, text processing programs, and more! Dr. Dobb's Journal offers **THE TOOLBOOK** in a special hardbound edition for only $29.95. You'll find:

- James Hendrix's famous Small C Compiler v. 2 and A New Library for Small C (also available on disk)
- Ron Cain's original A Small-C Compiler for the 8080's
- Never before published: Hendrix's new *Small-Mac: An Assembler for Small C* and *Small Tools: Programs for Text Processing* (both available on disk)
- Plus many useful programming tools in C

Also from M&T Publishing and Brady Communications—

The Small-C Handbook (Item #006)

The Small-C Handbook by James Hendrix is a valuable source of information about the Small-C compiler. In addition to descriptions of the language and the compiler, **The Handbook** also contains the entire source listings of the compiler and its library arithmetic and logical routines.

A perfect companion to the Hendrix Small-C compiler offered by Dr. Dobb's on disk, **The Handbook** even tells how to use the compiler to generate new versions of itself. $17.95

Dr. Dobb's C Tools On Disk

To complement **The Toolbook** Dr. Dobb's also offers the following programs on disk for only $19.95 each. Full source code is included and, except where indicated, both CP/M and MS or PC DOS versions are available.

Small-C Compiler (Item #007)

Jim Hendrix's **Small-C Compiler** is the most popular piece of software published in Dr. Dobb's 10-year history. Like a home study course in compiler design, the **Small-C Compiler** and **The Small-C Handbook** provide all you need to learn how compilers are constructed, as well as teaching the C language at its most fundamental level. **The Small-C Handbook** provides documentation for both versions ; however, an addendum is recommended in addition to **The Handbook** for MS or PC DOS-specific documentation. The addendum is available for $4.95. **(Item #008)**

Small Tools: Programs for Text Processing (Item #009)

This package consists of programs designed to perform specific functions on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying and concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts. This package includes only source code. Documentation available for $9.95. **(Item #010)**

Small-Mac: An Assembler for Small-C (Item #011)

Small-Mac is a macro assembler designed to stress simplicity, portability, adaptability, and educational value. The package features simplified macro facility, C-language expression operators, descriptive error messages, object file visibility, and an externally defined machine instruction table. Included programs are: macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and dump relocatable files. This program is available for CP/M systems only. Documentation available for $9.95. **(Item #012)**

To order by credit card call toll free: 1-800-528-6050 ext. 4001. Refer to the item number of each product. Or, send this order form with your payment to: **Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303.**

	Check Format: CP/M	Check Format: MS or PC DOS			QTY.		TOTAL
BOOKS							
Dr. Dobb's Toolbook			$29.95	X		=	
Small-C Handbook			$17.95	X		=	
DISKS							
Small-C Compiler			$19.95	X		=	
Small Tools Text Processor			$19.95	X		=	
Small Mac Assembler (for CP/M only)			$19.95	X		=	
Small-C Compiler MSPC-DOS specific addendum to The Small-C Handbook (NOTE: The Small-C Handbook provides full documentation for the CP/M version)			$4.95	X		=	
Small Tools			$9.95	X		=	
Small Mac			$9.95	X		=	

Sub Total $

California residents add applicable sales tax — Tax

Add $1.75 per item for shipping in U.S., $4.25 per item outside U.S. — Shipping

TOTAL $

For CP/M system disks only, please specify one of the following formats:

- ☐ Kaypro
- ☐ Apple
- ☐ Zenith Z- 1 00 DS/DD
- ☐ Osborne
- ☐ 8″ SS/SD
- ☐ Inquire about other formats

PAYMENT MUST ACCOMPANY YOUR ORDER

☐ Check enclosed

☐ Please charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card #____________________

Exp. Date____________________

Signature____________________

Name____________________

Address____________________
(please use street address)

City____________________

State__________Zip__________

Allow 6–12 Weeks for delivery

3109A

OF INTEREST

by Alex Ragen

CP/M

Electronic Business, a magazine devoted to the dollars and cents issues of the computer business, recently published a long article analyzing the microprocessor market. It carried the surprising message that 8-bit processors make up by far the lion's share of that market, with 16-bit processors trailing far behind. Moreover, most industry observers expect this situation to continue for many years. The reason is that the now inexpensive 8-bit processors and their peripherals are perfectly adequate for most applications. Where 16-bit processors hold sway is in the prestige conscious world of the personal computer, where only the latest fashion is marketable. In my corner of the world, everyone talks only about BMWs and IBM PCs, but they drive Toyotas and, when they build a piece of equipment, they use Z80s whenever they can and develop the software for it with that workhorse, CP/M 80, the reports of whose death have been greatly exaggerated.

It's true that finding a place to buy CP/M 80 software is a challenge, but there is plenty of good software out there. Those who persevere will find that their blood, sweat and tears will be rewarded. One of the most nagging problems a dealer has in stocking CP/M 80 software is that he has to carry several dozen disk formats. As a result stores are not a good place to find software. The best source is the manufacturer, who can whip up a disk for your format using one of the disk emulation programs now widely available. A good source for advertisements is Kaypro's magazine, *PROFILES*, which is available at most Kaypro dealers.

Echelon Inc. is one company that continues to develop quality CP/M 80 software. Its *pièce de résistance* is ZCPR3, the CCP replacement which gives CP/M 80 a number of Unix like features. Also SYSLIB3, a macro library with 210 subroutines, DISCAT, a disk and file utility program, TERM 3, a telecommunications program, and many others. All programs come with extensive documentation. For a catalog and price list, contact the vendor at 101 First St., Los Altos, CA 94022, (415) 948-3820. **Reader Service Number 101.**

The KAMAS "Outline Processor" is now available for CP/M 80 2.2. KAMAS combines outline processing and information retrieval with word processing to provide a complete environment for developing and controlling text. KAMAS is an aid for writers, researchers and other professionals who must organize their ideas and categorize information. It allows users to classify information in a familiar outline structure, and then to alter and access the information based on that structure. Levels of the outline can be collapsed from the screen and hidden from view, and then expanded back into view for editing. KAMAS includes a full screen editor with document output controls as well as a Forth-like programming language. There is a growing body of public domain add-ons, and the vendor offers three volumes of these utilities. The price is $147 plus $4 shipping. Contact KAMASOFT Inc., 2525 S.W. 224th Avenue, Aloha, OR 97006, (503) 649-3765. **Reader Service Number 103.**

Alternative Languages

The Forth-83 Handy Reference Card is now available free from the Forth Interest Group (FIG). It functions as a pocket programming aid and reference for Forth programmers. Commands are grouped by function and include: stack manipulation, comparison, arithmetic and others. For additional information about the card and FIG, call the FIG hot line at (408) 277-0668 or write FIG, P.O. Box 8231, San Jose, CA 95155. **Reader Service Number 105.**

MacScheme is a modern implementation of Lisp for the 512K Macintosh. It offers an interpreter with full run time error detection and debugging and adheres to the standards for Scheme, an influential dialect of Lisp. It supports the most important features of Lisp: lexically scoped variables, first class procedures (closures), macros, and generic arithmetic, including floating point and infinite precision integers. The Smalltalk-like interface features multiple scrolling windows. The price is $125 without copy protection. Contact Semantic Microsystems, 1001 Bridgeway, Suite 543, Sausalito, CA 94965. **Reader Service Number 111.**

UO-LISP, which contains over 400 standard Lisp functions and comes with an optimizing compiler that generates native 8086 code, is now available for the IBM PC and compatibles. There is a multi-window screen editor that supports Lisp interactions and a complete program development package. The entire package is priced at $150. The LEARN LISP system, a subset with on line help, a special reference manual and tutorial guide, is priced at $85. UO-LISP2, a CP/M version, costs $125. Contact Northwest Computer Algorithms, P.O. Box 90995, Long Beach, CA 90809, (213) 426-1893. **Reader Service Number 113.**

CLISP is a Lisp interpreter written in C for the IBM PC. It has 40 functions for list manipulation, arithmetic, relational and Boolean opera-

tions. A modest library of Lisp functions is also included. The price is $150, which includes source code and documentation. Contact WESTCOMP Software Development Group, 517 North Mountain Avenue, Suite 229, Upland, CA 91786, (714) 982-1738. **Reader Service Number 115.**

Useful and practical expert systems can be designed and implemented by users with no prior knowledge of AI programming with the ES/P Advisor, according to the vendor. The product, written in Prolog 2 for the IBM PC, gives the user the benefit of general purpose (and Prolog 2) interfacing, fast execution, windows, garbage collection and other features. There are no limits in knowledge base or number of rules that can be handled. The price is $895. Contact Expert Systems International, 1150 First Avenue, King of Prussia, PA 19406, (215) 337-2300. **Reader Service Number 117.**

STOK PILOT-PC, a language for the development of advanced training and control applications, is now available for the IBM PC. This is the only high level language with integrated instructions for the storage and manipulation of audio on a hard disk. Contact Stok Software, 17 West 17th St., New York, NY 10011, (212) 243-1444. **Reader Service Number 119.**

Super Pascal, a complete development system for the Commodore 64 and 128 computers, is available for $59.95 from Abacus Software, 2201 Kalamazoo S.E., P.O. Box 7211, Grand Rapids, MI 49510, (616) 241-5510. **Reader Service Number 121.**

Apple

A transportable 300 baud modem which clips on to the back of the Apple IIc and solves the problem of sudden disconnect when a call comes in on a "call waiting" line, has been announced by Prometheus Products. The price is $199.95. Contact the vendor at P.O. Box 4156, Fremont, CA 94539, (415) 490-2370. **Reader Service Number 123.**

The Mac 3.5 card provides Apple II users with a 3.6 MHz 65C02 processor that speeds up their Apple in some cases by as much as 3.5 times,

though the vendor is careful to point out that for programs that contain most of their code in the auxiliary memory of the Apple IIe, e.g. Pascal 1.2, the speed advantages will be less impressive. Contact A.P.P.L.E Co-op, 290 S.W. 43rd St., Renton, WA 98055, (206) 251-5222. **Reader Service Number 125.**

A RAM disk for the Macintosh is available from Symmetry Corporation. Quickdisk requires a 512K Mac or XL and costs $34. Contact Symmetry at 3900 East Camelback Road, Suite 103-S, Phoenix, Arizona 85018 (602) 224-5944. **Reader Service Number 127.**

Hippopotamus Software has announced the availability of level 2 of its Hippo-C for the Macintosh. Level 1, which has been available for several months, is designed for use by educational institutions and the general public. It incorporates the familiar Macintosh interface and includes an editor, full C compiler, linker, symbolic debugger, standard C library, online tutorial, and shell command processor. Level 2 features an optimizing compiler, 68000 assembler, linker, librarian, full floating-point support, header files and a Unix-like command shell, access to over 500 toolbox routines, the sound channels, and serial ports. Level 1 retails for $149.95, and level 2 retails for $399.95. An upgrade kit for level 1 costs $250. There are no licensing fees for software produced using Hippo-C, and nonprotected versions are available for $25 more. **Reader Service Number 129.**

The same vendor also has introduced Hippo-Lock, a "business Data Security System" for the Mac, which scrambles the contents of a file once the user creates a keyword phrase using the DES standard. Hippo-Lock works with microdisks, hard disks, and over the AppleTalk office network, and it can be used to scramble electronic mail on MCI mail, EasyLink, and Compuserve. Contact the vendor at 985 University Avenue, Suite 12, Los Gatos, California 95030 (408) 395-3190. **Reader Service Number 131.**

DeskTop Software has announced a solution for the problem of incompatibility of copy protection schemes and hard disk usage. The new system, which transfers the copy protection information to the hard disk, is available for all three of DeskTop's products: 1stBASE, 1stPORT, and 1stMERGE. Present users can obtain the new system at no charge. Contact the vendor at 244 Wall Street, Princeton, New Jersey 08540 (609) 924-7111. **Reader Service Number 133.**

IBM-PC

The Disc Interchange Service converts files written in one format to a different format. They have added the 3½ inch drive to their format conversion capabilities, bringing the number of formats supported to nearly 200. The following formats are available: DG/One, Kaypro 2000 and the IBM PC 2 (when its details become known), as well as the Hewlett-Packard HP-150. Contact the vendor at 481 Great Road, Acton, MA 01720, (617) 263-6001. **Reader Service Number 109.**

A software package which is designed to ensure the security of data in a shared computer environment without the use of encryption has been announced by AST Research. The Knight Data Security Manager offers passwords, copy protection utilities and a variety of file management features for the IBM PC and compatibles. A complete menu driven DOS interface is also provided, though access to DOS is still possible in the normal way. Contact the vendor at 2121 Alton Ave., Irvine, CA 92714, (714) 863-1333. **Reader Service Number 135.**

X-VIEW 86, which allows hardware and software experts to observe the internal operations of DOS applications software, has been introduced by McGraw Hill. It is available for immediate shipment and is priced at $59.95 plus shipping. X-VIEW 86 is a software analyzer and consists of a diskette, documentation and technical support for registered users. the product is available on a special 10 day examination basis, and the price is fully refundable, provided the protective seal on the diskette is unbroken. Contact the vendor at 8111 LBJ Freeway, Dallas, TX 75251, (800) 221-8439, or (800) 233-8439 in Texas. **Reader Service Number 137.**

CP/EM makes CP/M 80 emulation in software available for the IBM PC and compatibles. It is priced at $79.95. Contact the ICU Group, P.O. Box 10118, Rochester, NY 14610, (716) 425-2519. **Reader Service Number 139.**

C and Unix

Computer Innovations has announced version 2.3 of their Optimizing C86 compiler, whose most important new feature is source level debugging support. This allows popular debuggers such as Periscope, Pfix Plus and Atron to be used. The documentation is completely new. It is priced at $395, or $35 for an upgrade for exsiting C86 users. Contact the vendor at 980 Shrewsbury Avenue, Tinton Falls, NJ 07724, (800) 922-0169. **Reader Service Number 141.**

HEXPOSE is a "visual" binary file editor for Unix which can be used for patching object modules, repairing damaged files and verifying the results of I/O operations. For further information, contact Specialized Systems Consultants, P.O. Box 55549, Seattle, WA 98125, (206) 367-UNIX. **Reader Service Number 143.**

National Semiconductor has announced the production release of Genix 4.2, a port of Berkeley 4.2 BSD Unix for National's Series 32000 32 bit microprocessor family. The package includes a C compiler and porting tools for a VAX/4.2 host. A Pascal compiler is available as well. National also offers System V/ Series 32000, based on AT&T's Unix System V. Contact the vendor at 2900 Semiconductor Drive, Santa Clara, CA 95051 (408) 721-5000. **Reader Service Number 145.**

Software Architecture and Engineering has announced a line of embeddable expert system tools implemented in C. Initial releases of the Knowledge Engineering System II support the IF . . . THEN production rules for inferencing, and future releases will add frame-based pattern matching and statistical inferencing methods. Contact the vendor at 1500 Wilson Boulevard, Suite 800, Arlington, Virginia 22209 (703) 276-7910. **Reader Service Number 147.**

Software Research Associates has introduced its Test Coverage Analysis Tool (TCAT) for C, which the vendor claims will improve the quality of unit and system level testing by a factor of five or more. It identifies which parts of a program have been tested and which have not. TCAT/C is available for VAX/Unix, VAX/VMS, and IBM PC and compatibles. Prices range from $975 to $9000 depending on the configuration. SRA is prepared to port TCAT/C to almost any operating system on a custom basis. For further information, contact the vendor at 580 Market Street, San Francisco, California 94104 (415) 957-1441. **Reader Service Number 149.**

A training package for C has been announced by Computer Innovations, the developers of C-86. "Introducing C," which consists of a self-paced training manual and C interpreter, runs on an IBM PC (DOS 2.0) with 192K. The price is $95. Contact the vendor at 980 Shrewsbury Avenue, Tinton Falls, New Jersey 07724. **Reader Service Number 151.**

Smart/C is an integrated precompilation development environment for C; it consists of a syntax-directed editor integrated with a screen-oriented interpreter and the Migrator, which allows existing C programs to become "amenable" to the Smart/C environment. It runs under Unix System V, Berkeley BSD 4.2, Xenix, and MSDOS. Contact AGS Computers, Advanced Products Division, 1139 Spruce Drive, Mountainside, New Jersey 07092 (201) 654-4321. **Reader Service Number 153.**

Interactive/C is a full K&R development system that includes a command processor, full screen editor, source level debugger, and execution profiler. Its multiwindow multi-user interface permits debugging of full screen graphics on one or more CRTs with simultaneous display of source code, program output, and system status. It is compatible with Lattice C and lists for $395. Contact the vendor, IMPACC Associates, at P.O. Box 93, Gwynedd Valley, Pennsylvania 19437. **Reader Service Number 155.**

C-LINK is an application generator that, working with the vendor's S-Tran BASIC to C translator, allows a programmer to develop programs in C by writing in BASIC. C-LINK runs under Unix or Xenix and costs $695. For further information, contact the vendor, SMI, at 20720 South Leapwood Avenue, Carson, California 90746 (213) 538-8174. **Reader Service Number 157.**

DDJ

ADVERTISER INDEX

* This advertiser prefers to be contacted directly: see ad for phone number.

Advertising Sales Offices

East Coast
Walter Andrzejewski (617) 567-8361

Midwest
Michele Beaty (317) 875-8093

Northern California/Northwest
Lisa Boudreau (415) 424-0600

Southern California/AZ/NM
Beth Dudas (714) 643-9439

Advertising Director
Shawn Horst (415) 424-0600

New lightning-fast ZBASIC zaps the competition.

It's hot. It's brand-new. And light years ahead of anything else.

It's ZBASIC. Written *for* programmers *by* programmers. (If you know BASIC—you know ZBASIC!) Now you can write a program exactly the same way on an Apple, an IBM, a Tandy, or any other major micro and port the source code. You only write the program *once* ...and it runs on all the major micros. The commands stay the same—regardless of the computer* (even graphic commands and disk I/0!).

ZBASIC. Starting now, it's the only language you'll ever have to know.

*subject to hardware limitations.

AVAILABLE NOW!

- Works the same on all micros
- Uses same commands—regardless of computer make
- Structured Basic—(spaghetti optional)
- Device-independent graphics (same graphic commands on all computers)
- 6 – 54 digits of precision (selectable by user)
- Built-in interactive editor and compiler—to compile and execute, just type "run."
- Choice of alphanumerics labels or line numbers
- Chaining with shared variables

Not copy protected—No Run Time Fees or Royalties
One low price gives you everything—there are no hidden costs. Only $89.95 complete.

The finest implementation of the BASIC language for microcomputers!

•=YES N/A=Not available	ZBasic Interpiler 3.0	TURBO PASCAL 3.0	MBASIC compiler	True BASIC	BASICA Interpreter (IBM PC)
IBM and compatibles	•	•	•	•	•
Apple //e, //c (6502)	•	N/A	N/A	N/A	N/A
Macintosh	4th Qtr.	N/A	N/A	N/A	N/A
CP/M-80 2.2, 3.0	•	•	•	N/A	N/A
TRS-80 Mod I, III, 4, 4p	•	N/A	•	N/A	N/A
Direct commands	•	N/A	N/A	•	•
Maximum scientific digits of accuracy. (COS, SIN, ATN, LOG, EXP etc.)	6 to 54 selectable by the user	11 Binary BCD N/A	16	16	6
Device Indpdnt Graphics (same CMDS all graphic modes and computers)	•	N/A	N/A	N/A	N/A
SAME File commands all computers?	•	N/A	N/A	N/A	N/A
STRUCTURED: Labels, Functions, LONG IF etc.	•	•	N/A	•	N/A
Same editor commands all versions/computers	•	•	N/A	N/A	N/A
Sieve benchmark (Byte January 1983, 10 iter's)	13.7 sec.	14.1 sec.	14.9 sec.	261 sec.	2190 sec.
Shell-Metzner SORT (Sybex-BASIC for Scientist's and Eng. 2,000 5 char. strings)	19 sec.	28 sec.	71 sec.	194 sec.	2700 sec.
Executable Machine Lang. & approx. File size	12k	12k	32k	N/A	N/A
PRICE with BCD BCD=No rounding errors)	89.95	109.95	450.00	N/A	N/A
PRICE without BCD	89.95 BCD FREE	69.95	395.95	149.95	Comes with computer

ZBASIC™

The lightening-fast BASIC from Zedcor

To order use this coupon or call

ORDERS ONLY: 1-800-482-4567

SCHOOLS – Call For Special Package

Utah Residents 1-800-662-8666. Alaska Residents 1-800-982-1500

INQUIRIES : (602) 795-3996

Mail to: ZEDCOR
3438 N. Country Club Road / Tucson, AZ 85716

(Name)

(Address) (Apt.)

(City) (State) (Zip)

(Day Phone)

☐ ZBASIC IBM PC/ Compatible (128K, MS DOS 2.0 or better recommended)	$89.95	$______
☐ ZBASIC Apple //e, //c (128K, Dos 3.3) (Integrate text & graphics)	$89.95	$______
☐ ZBASIC CPM-80 (Z-80-2.2 or 3.0) (Provided on 5¼" KAYPRO II-SSDD Format)	$89.95	$______
☐ KAYPRO graphics Version (11, 4, 10)	$89.95	$______
☐ ZBASIC TRS-80 (48K) (Circle Model 1,3 or 4-6.2) (Hi Res Bd's Supported)	$89.95	$______
☐ ZBASIC Macintosh (Delivery 4th quarter)	$89.95	$______
☐ DEVELOPERS— BUY ALL 6, SAVE $140. $449.75 VALUE	$399.75	$______
Arizona Residents Add 5% Sales Tax		$______
SHIPPING: U.S. add $5.00 per program		$______
Foreign or C.O.D. add $10.00 per program (U.S. currency only)		$______
TOTAL		$______

All benchmarks and accuracies apply to standard IBM PC with 8088. Other computers and/or CPU benchmark speeds, accuracies and object code file sizes will vary depending on the computer clock speed, operating systems and other factors beyond our control.

Circle **no. 160** on reader service card.

SAVE OVER 30% ON OUR GIFT PACKS!
60-DAY MONEY-BACK GUARANTEE

How Borland's Three New Holiday Packs Will Fill Your Stocking Without Emptying Your Piggybank.

Three special packs with dazzling discounts that will help get you into a Holiday mood. You can get some of Turbo, most of Turbo, or all of Turbo—including the two newest members of the Turbo family, Turbo GameWorks™ and Turbo Editor Toolbox.™ You also get our unmatched 60-day money-back guarantee, quality products that aren't copy-protected.

TURBO NEW PACK $95.00.

You get the two exciting new members of the Turbo Pascal family,

- TURBO GAMEWORKS, Chess, Bridge, and Go-Moku, complete with source code and a 200-page manual.
- TURBO EDITOR TOOLBOX, all the building blocks to make your own editors and word processors, complete with source code and a 200-page manual.

TURBO HOLIDAY PACK $125.00.

You get all three of the Turbo family classics for only $125.00 (about a 30% discount). Turbo Pascal 3.0 and Turbo Tutor and Turbo DataBase Toolbox—all for just $125.00.

- TURBO PASCAL combines the fastest Pascal compiler with an integrated development environment.
- TURBO TUTOR teaches you step-by-step how to use Turbo Pascal with commented source code for all program examples on diskette.
- TURBO DATABASE TOOLBOX offers three problem-solving modules for your Turbo Pascal programs: Turbo Access, Turbo Sort, and GINST, which generates a ready-to-run installation program that lets you forget about adapting your software to specific terminals.

TURBO HOLIDAY JUMBO PACK $245.00.

This is it—the whole thing, the entire Turbo family including its two newest members. You get:

- Turbo Pascal
- Turbo Graphix Toolbox
- Turbo Tutor
- Turbo DataBase Toolbox
- Turbo GameWorks
- Turbo Editor Toolbox

NEW!

and you pay only $245.00 for all six! Which means that you're getting everything at only about $40 a piece. Quite a holiday deal. (And if you already own one or several members of the Turbo family, be creative—nothing can stop you from buying the Jumbo Pack, picking out the ones you already have and giving the rest as holiday gifts to family and friends. At these prices you can afford to give to others and to yourself.) Speaking of Holidays, this offer lasts until March 31, 1986. (At Borland, we like to make the Holidays last.)

NEW!

TURBO GAMEWORKS $69.95.

Our new Turbo GameWorks offers games you can play and replay without Turbo Pascal or revise and rewrite with Turbo Pascal 3.0. We give you the source code, the manual, the diskettes and the competitive edge. Chess, Bridge and Go-Moku. State-of-the-art games that let you be player, referee, and rules committee all at once because you have the Turbo Pascal source code. Learn exactly how the games are made—so you can go off and make your own. And Turbo GameWorks is the only quality game you can buy that is not copy-protected. Sold separately, only $69.95. (Just $47.50 if you buy the Turbo New Pack.)

NEW!

TURBO EDITOR TOOLBOX $69.95.

Build your own word processor—for only $69.95! You get ready-to-compile source code, a full-featured word processor that looks and acts like WordStar™, and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs. With Turbo Editor Toolbox you can have the best of all word processors. You can make WordStar behave like Multi-Mate. Support windows just like Microsoft's Word. And do it as fast as WordPerfect does it. Incorporate your new "hybrids" into your programs to achieve incredible control and power. Sold separately, only $69.95. (If you buy the Turbo New Pack, the price drops to just $47.50.)

NOT COPY-PROTECTED

Holiday Gift Packs, Turbo GameWorks™ & Turbo Editor Toolbox™

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133.

The holiday packs include an upgrade coupon for both options so you get BCD and 8087 support for $39.95 (regularly $55.00).

*Gift Pack Offers Last Until March 31, 1986

		Quantity
*Turbo Holiday Jumbo Pack	$245.00	____
*Turbo Holiday Pack	$125.00	____
*Turbo New Pack	$95.00	____
Pascal	$69.95	____
Pascal w/8087	$109.90	____
Pascal w/BCD	$109.90	____
Pascal w/8087 and BCD	$124.95	____
Turbo DataBase	$54.95	____
Turbo Graphix	$54.95	____
Turbo Tutor	$34.95	____
Turbo Editor	$69.95	____
Turbo GameWorks	$69.95	____

These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Carefully describe your computer system!

Mine is: __ 8-bit __ 16-bit

I use: __ PC-DOS __ MS-DOS __ CP/M-80 __ CP/M-86

My computer's name and model is: ____________

The disk size I use is: ☐ 3½" ☐ 5¼" ☐ 8"

Name: ____________

Shipping Address: ____________

City: ____________

State: ____________ Zip: ____________

Telephone: ____________

H11

NOTE: Turbo Editor Toolbox and Turbo GameWorks are available for the IBM PC and true-compatibles using Turbo Pascal 3.0 ONLY.

Amount: (CA add 6% tax) ____________

Payment: VISA MC Check Bank Draft

Credit Card Expiration Date: ______ / ______

Card # ____________

COD's and Purchase Orders will not be accepted by Borland International. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in U.S. dollars drawn on a U.S. bank.

BORLAND INTERNATIONAL

4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

Circle no. 14 on reader service card.